Universidad Autónoma de Baja California

Facultad de Ingeniería

Design and Implementation of an Automatic and Self-Adaptive NILM System Using Deep Learning and an IoT Platform

Maestría y Doctorado en Ciencias e Ingeniería

THESIS

For the Degree of

Doctor of Engineering

By

Omar Muñoz Urías

Thesis Advisor

Dr. Adolfo Heriberto Ruelas Puente

Mexicali, Baja California, México

October 2025

Dedicated to
My parents, my grandparents,
and my beloved partner.

Acknowledgments

To Dr. Adolfo Heriberto Ruelas Puente, for his insightful guidance and dedication to this project, and above all, for his support, valuable advice, and friendship.

To the members of the thesis committee, Dr. Carlos Alberto Aguilar Avelar, Dr. Jorge Eduardo Ibarra Esquer, Dr. Pedro Francisco Rosales Escobedo, and Dr. Gabriel Ernesto Pando Martinez for their valuable feedback and willingness to contribute.

To the Facultad de Ingeniería, Universidad Autónoma de Baja California, Mexicali campus.

To my family, for being an unwavering source of support at all times, and above all, for understanding the circumstances involved in carrying out a graduate study of this magnitude.

To the SECIHTI National Scholarship Program.

Abstract

Non-Intrusive Load Monitoring (NILM) has emerged as a promising approach to enable smarter grids and promote efficient electricity use. Recent advances in the Internet of Things and Artificial Intelligence have accelerated the development of NILM solutions; however, practical implementations that achieve real-time operation while maintaining scalability and adaptability in diverse residential settings remain scarce. To address this gap, this work presents the design and implementation of a real-time IoT-based NILM system for residential environments, integrating event detection, load disaggregation, and automated dataset generation. The system employs an event-based methodology comprising four stages: data acquisition, event detection, feature extraction, and load recognition. A custom smart meter, equipped with an embedded event detection and disaggregation algorithm, captures aggregated household load signals and extracts appliance-level signatures. Data is transmitted via the MQTT protocol to a cloud-based platform for storage, visualization, and further processing. An autoencoder model based on 1D Convolutional Neural Networks combined with K-means clustering achieved a 99.23% accuracy in grouping appliance events into their correct categories, while unsupervisedly enabling the automatic creation of a tailored dataset without relying on public datasets. The generated dataset is subsequently used to train three deep learning models for time series classification, yielding overall accuracies of 98.53% with a 1D Convolutional Neural Network, 97.91% with a Long Short-Term Memory network, and 98.28% with a WaveNet model. A web application was developed to visualize detected events and classification results, achieving an average end-to-end response time of 1.05s. Experimental validation in two different residential environments demonstrated robustness, scalability, and classification accuracy exceeding 95% across diverse consumption profiles, confirming the feasibility and efficiency of the proposed self-adaptive NILM system for real-world energy monitoring applications.

Resumen

El Monitoreo No Intrusivo de Cargas (NILM, por las siglas en inglés Non-Intrusive Load Monitoring) ha surgido como un enfoque prometedor para habilitar redes eléctricas más inteligentes y fomentar un uso eficiente de la electricidad. Los avances recientes en el Internet de las Cosas y la Inteligencia Artificial han acelerado el desarrollo de soluciones NILM; sin embargo, las implementaciones prácticas que logran operación en tiempo real, manteniendo al mismo tiempo escalabilidad y adaptabilidad en entornos residenciales diversos, siguen siendo limitadas. Para atender esta brecha, este trabajo presenta el diseño e implementación de un sistema NILM en tiempo real basado en IoT para entornos residenciales, integrando detección de eventos, desagregación de cargas y generación automática de conjuntos de datos. El sistema emplea una metodología basada en eventos compuesta por cuatro etapas: adquisición de datos, detección de eventos, extracción de características y reconocimiento de cargas. Un medidor inteligente personalizado, equipado con un algoritmo embebido de detección de eventos y desagregación, captura señales de carga agregada del hogar y extrae señales características a nivel de electrodomésticos. Los datos se transmiten mediante el protocolo MQTT a una plataforma en la nube para almacenamiento, visualización y procesamiento adicional. A los eventos detectados se les aplica un modelo de autoencoders con redes neuronales convolucionales unidimensionales, combinado con el algoritmo de agrupamiento K-means, lo que posibilita la creación automática de un conjunto de datos personalizado. De esta forma se alcanzó una precisión del 99.23% en la clasificación de los eventos dentro de sus categorías correspondientes, eliminando así la dependencia de bases de datos públicas. Al conjunto de datos generado se empleó posteriormente para entrenar tres modelos de aprendizaje profundo para clasificación de series temporales, obteniendo precisiones globales de 98.53% con una red neuronal convolucional unidimensional, 97.91% con una red de memoria larga a corto plazo y 98.28% con un modelo WaveNet. Se desarrolló una aplicación web para visualizar los eventos detectados y los resultados de clasificación, alcanzando un tiempo promedio de respuesta extremo a extremo de 1.05s. La validación experimental en dos entornos residenciales distintos demostró robustez, escalabilidad y precisiones de clasificación superiores al 95% en perfiles de consumo diversos, confirmando la viabilidad y eficiencia del sistema NILM auto-adaptativo propuesto para aplicaciones reales de monitoreo energético.

Contents

List of Figures **xi**

List of Tables **xiv**

1 Introduction **1**

1.1 State-of-the-art . . . 3
1.2 Problem Statement and Justification . . . 14
1.3 Research Hypothesis . . . 17
1.4 Research Objectives . . . 17
1.4.1 General Objective . . . 17
1.4.2 Specific Objectives . . . 17
1.5 Methodology . . . 18

2 Theoretical Principles **22**

2.1 Load Monitoring Concept . . . 22
2.1.1 Intrusive Load Monitoring . . . 23
2.1.2 Non-intrusive Load Monitoring . . . 23
2.1.3 Real-time Requirements in NILM Systems . . . 24
2.2 Analysis of Electrical Parameters in NILM . . . 25
2.2.1 Steady State Features . . . 26
2.2.2 Transient State Features . . . 26
2.2.3 Sampling Rate . . . 27
2.2.4 Classification of Loads . . . 28
2.3 Electrical Parameters at High Frequency Sampling Rate . . . 29
2.3.1 Raw Current Signal . . . 30
2.3.2 V-I Trajectories . . . 30
2.3.3 Harmonic Components . . . 32
2.3.3.1 Fourier Analysis . . . 33

2.4 Electrical Parameters at Low-Frequency Sampling Rate 33
2.4.1 Power 33
2.4.2 Instantaneous Power 34
2.4.3 Active Power 34
2.4.4 RMS Voltage and Current 35
2.4.5 Apparent Power 36
2.4.6 Reactive and Complex power 36
2.4.7 The Power Triangle 37
2.5 Artificial Intelligence, Machine Learning and Deep Learning 38
2.5.1 Artificial intelligence 38
2.5.2 Machine Learning 39
2.5.3 Deep Learning 41
2.6 Overview of Machine Learning and Neural Network Models 42
2.6.1 K-Means Clustering 42
2.6.2 Artificial Neural Networks 43
2.6.2.1 The Perceptron: Input-Output Structure 44
2.6.2.2 ANN Structure and Layers 44
2.6.2.3 Activation Functions 45
2.6.2.4 Backpropagation 46
2.6.3 1D Convolutional Neural Network 47
2.6.3.1 Structure of a 1D CNN 47
2.6.3.2 Convolution Operation 48
2.6.3.3 Padding 48
2.6.3.4 Pooling Layers 49
2.6.3.5 Fully Connected Layers 49
2.6.4 Recurrent Neural Networks 49
2.6.4.1 RNN input layer 50
2.6.4.2 Hidden State and Recurrent Connections 50
2.6.4.3 Output Layers 51
2.6.4.4 Unfolding Mechanism 51
2.6.4.5 Backpropagation Through Time (BPTT) 51
2.6.5 Long Short Term Memory networks 52
2.6.5.1 Memory Cell and Cell State 52
2.6.5.2 Forget Gate 53
2.6.5.3 Input Gate and Candidate Cell State 53
2.6.5.4 Output Gate and Hidden State 53

2.6.6 Evaluation Metrics 54
2.7 Internet of Things (IoT) 55
2.7.1 IoT Architecture 56
2.7.1.1 Perception Layer 57
2.7.1.2 Network Layer 57
2.7.1.3 Processing Layer 58
2.7.1.4 Application Layer 58
2.7.1.5 Business Layer 58
2.7.2 Technologies in the Perception Layer 58
2.7.2.1 Wireless Sensor Networks (WSN) 58
2.7.3 Technologies in the Network Layer 59
2.7.3.1 Communication Models 59
2.7.3.2 Communication Protocols 60
2.7.3.3 Link Layer 60
2.7.3.4 Network Layer 62
2.7.3.5 Transport Layer 62
2.7.3.6 Application Layer 63
2.7.4 Technologies in Middleware Layer 64
2.7.4.1 Cloud Computing 64

3 System Design **66**
3.1 Requirements and Purpose Specifications 66
3.2 Process Specification 67
3.3 Domain Model Specification 69
3.3.1 Physical Entity 70
3.3.2 Virtual Entity 71
3.3.3 Devices 71
3.3.4 Resources 71
3.3.5 Services 72
3.4 IoT Level Specification 72
3.4.1 Single-Node Architecture with Cloud-Centric Analysis 73
3.4.2 Cloud-Centric Processing and Scalability 74
3.5 Functional View Specification 74
3.6 Operational View Specification 76

4 System Implementation **78**
4.1 NILM Architecture . . . 78
4.2 Data Acquisition Device . . . 79
4.3 Event detection algorithm . . . 80
4.4 Automatic and Adaptive Dataset . . . 82
4.4.1 Feature Extraction with 1-D CNN Autoencoders . . . 83
4.4.2 Clustering with Automatic k Centroids . . . 83
4.5 Supervised Classification Models . . . 84
4.5.1 1D-CNN Implementation . . . 85
4.5.2 WaveNet Model . . . 86
4.5.3 LSTM Model . . . 87
4.6 Web Application . . . 88
4.6.1 Backend . . . 88
4.6.2 Frontend Interface . . . 89

5 Experiments and Results **92**
5.1 Experimental Setup Data Collection . . . 92
5.2 Electrical Parameters Selection . . . 93
5.3 Detected events . . . 94
5.4 Feature Extraction via Autoencoder . . . 97
5.5 Unsupervised Clustering and Dataset Creation . . . 97
5.6 Classification with Deep Learning Models . . . 101
5.7 Computational Performance . . . 107
5.8 Dataset Adaptability in Supervised Learning Model . . . 108
5.9 Cross-Household Generalization . . . 108
5.10 Event Classification Web Response Time . . . 112

6 Conclusions and Future Work **115**
6.1 Conclusions . . . 115
6.2 Future Work . . . 117

A Academic Products **130**
A.1 Article Title: *Design and implementation of an automatic and self-adaptive NILM system using unsupervised learning and an IoT platform* 130
A.2 Article Title: *Development of an IoT smart energy meter with power quality features for a smart grid architecture* . . . 131

A.3 Article Title: *Design and Development of an IoT Smart Meter with Load Control for Home Energy Management Systems* 132
A.4 Proceedings Article Title: *A WaveNet-Based IoT System for Non-Intrusive Load Monitoring* 133

B Datasets **134**
B.1 House 1 and House 2 signatures 134

C Smart Meter **135**
C.1 PCB Design 135
C.2 PCB Design 136
C.3 Firmware 136

Appendices **130**

List of Figures

1.1 Waterfall methodology steps. 18
1.2 Workflow for Machine Learning Development. 19
1.3 Methodology for NILM projects. 21

2.1 Representation of a NILM system. 24
2.2 Example of transient state and steady state. 27
2.3 Example of raw current waveform. 30
2.4 V-I trajectories of four loads from the COOL dataset: (a) Drill at full speed (750 W), (b) LED lamp (1.6 W), (c) Drill at half speed (750 W), and (d) Halogen lamp (105 W). 31
2.5 Harmonic components of a current waveform. 32
2.6 Power triangle. 38
2.7 Relationship between artificial intelligence, machine learning and deep learning. 39
2.8 Categories of machine learning. 40
2.9 Example of deep learning flow. 41
2.10 Demonstration of K-means application. 42
2.11 Perceptron structure. 44
2.12 Artificial Neural Network basic architecture. 45
2.13 Convolutional Neural Network basic architecture. 47
2.14 Recurrent Neural Networks basic architecture. 50
2.15 Long Short-Term Memory cell. 52
2.16 Five-layer architecture of the Internet of Things 57

3.1 Flowchart of how the smart meter works. 68
3.2 Flowchart of how measurements and events are treated inside the web server. 69

3.3 Domain model diagram representing the main concepts, entities, and interrelationships within the NILM IoT system. 70
3.4 IoT description of the NILM system. 73
3.5 Functional view specification diagram. 74
3.6 Operational view specification diagram. 76

4.1 Proposed system architecture. 79
4.2 Printed circuit board and enclosure. 80
4.3 Event detection process. 82
4.4 1-D CNN autoencoder architecture 84
4.5 1-D CNN architecture for classification. 85
4.6 WaveNet model used for classification tasks. 87
4.7 LSTM-based model used for appliance classification. 88
4.8 Backend architecture of the NILM system. 90
4.9 Main Dashboard tab showing real-time monitoring components: View A (gauge meters for kWs per phase), View B (area chart of daily active power), and View C (bar chart of monthly energy consumption). 90
4.10 NILM tab displaying appliance-level signal waveforms and monitoring table: View D (RMS current chart), View E (active power chart), View F (reactive power chart), View G (recognized appliances), View H (ON/OFF status), View I (last activation time), and View J (last deactivation time). 91

5.1 Testing environment: Smart meter installation and appliances used. . . 94
5.2 Weights of the electrical parameters in the first principal component. . 95
5.3 Normalized active power trajectories during ON and OFF events. . . . 96
5.4 Representation of the dataset structure for ON and OFF events. 96
5.5 Comparison between the original event and reconstructed event from latent space using 1-D CNN Autoencoders. 98
5.6 PCA visualization of latent representations from ON and OFF events of the seven appliances and the corresponding K-means clustering results. 99
5.7 Effect of increasing the number of appliances on the automatic selection of the number of clusters. 100
5.8 Performance of how automatic K cluster selection reflects the actual number of appliance combinations. 101
5.9 Training behavior of the 1D-CNN model in terms of loss and accuracy. 102
5.10 Classification performance of the 1D-CNN model. 102

5.11 Training behavior of the LSTM model in terms of loss and accuracy. . . 103
5.12 Classification performance of the LSTM model. 104
5.13 Training behavior of the WaveNet model in terms of loss and accuracy. 104
5.14 Classification performance of the WaveNet model. 104
5.15 Comparison of LSTM, 1D-CNN, and WaveNet Models in a 5-Fold Cross-Validation Test. 105
5.16 Effect of varying the percentage of samples per class on the classification accuracy of the 1D-CNN, LSTM, and WaveNet models. 106
5.17 Classification performance of the LSTM model across the 120 appliance combinations. 109
5.18 ON events generated in House 2 for the seven appliances, corresponding to their active and reactive power signatures. 110
5.19 Clustering of ON and OFF events using k-means with automatic cluster selection, visualized through a two-dimensional PCA projection. 110
5.20 Transient ON-event signatures computed from PLAID data after preprocessing the raw voltage and current signals into RMS current, active power, and reactive power. 112
5.21 Confusion matrix of the LSTM model evaluated on the PLAID dataset (CMU Lab subset). 113

A.1 Cover page of the article published in *Electric Power Systems Research* (Elsevier). DOI: 10.1016/j.epsr.2024.111376 130
A.2 Cover page of the article published in *Sustainable Computing: Informatics and Systems* (Elsevier). DOI:10.1016/j.suscom.2024.100990 131
A.3 Cover page of the article published in *Sensors* (MDPI). DOI: 10.3390/s22197536 132
A.4 Certificate of paper acceptance and publication for the article presented at the 10th International Conference on Information and Communication Technology for Intelligent Systems (ICTIS 2025), New York, USA. The paper was included in the book *ICT for Intelligent Systems: Proceedings of ICTIS 2025, Volume 13*, ISBN: 978-981-95-1352-9. 133

C.1 Schematic diagram of the designed IoT smart meter. 135
C.2 PCB of the designed IoT smart meter. 136

List of Tables

1.1 Comparison among existing NILM works. 14

5.1 Technical information of the appliances used in the experiments. 93
5.2 Classification performance of the tested models. 105
5.3 Computational performance comparison of the tested models. 107
5.4 Clustering and classification results for House 2, including LSTM performance with cross-validation and k-selection accuracy. 111
5.5 Event detection and classification response times. 114

Chapter 1

Introduction

As time goes by, humans are becoming increasingly connected to their surrounding environment, thanks to advancements in wireless technologies and internet connectivity. While devices used to communicate solely with each other, people have now become a fundamental part of this connection. This is where the concept of the Internet of Things (IoT) emerges. This approach involves integrating a series of "smart" devices into the environment to interact directly with living beings, either by providing services or delivering information, with the aim of improving quality of life [1, 2].

The Internet of Things has had a significant impact across many areas, ranging from healthcare to the energy sector, with the latter being the primary focus of this work. As mentioned previously, human connectivity continues to grow each day, primarily driven by the use of electronic devices that require electrical power to operate. These include mobile phones, computers, televisions, electric cars, and more. Likewise, as the global population increases, the adoption of such technologies also rises, resulting in an ever-growing demand for electrical energy. This surge in energy consumption brings various consequences across multiple areas, which will be discussed in more detail in subsequent sections of this document.

The increase in electrical consumption brings a series of challenges that can be addressed through greater energy efficiency, which is driven by the development of Smart Grids (SGs). These grids enable a balance between supply and demand through effective management based on the use of contemporary technologies [3, 4, 5, 6].

Traditional power grids have significant limitations, as they lack the ability to prevent or respond to sudden failures within the infrastructure. This is primarily due to their unidirectional nature, where the utility company does not receive timely or effective feedback regarding incidents [7, 8]. In contrast, SGs, with their bidirectional communication between production and demand, enable better coordination. For

instance, in a traditional structure, when the required energy exceeds the available supply, the utility company often resorts to complete service interruptions in areas of low commercial interest. However, with SGs, real-time information from end users allows for the identification of strategic areas where such actions can be taken more selectively and where they should be avoided [3].

Within Smart Grids, one of the main research topics—and the foundation of this work—is the monitoring and identification of electrical loads at the individual appliance level. The primary purpose of this is to facilitate energy conservation through the adoption of energy-efficient initiatives, such as using low-power appliances, optimizing the timing of device usage, and eliminating unwanted energy consumption activities [9]. Recently, there has been an increase in microgrids and a steady rise in the installation of renewable energy sources. To enhance the quality of these efforts, further strategies for energy sustainability must be implemented to monitor, control, and manage the energy system effectively. Load monitoring and automation enable Home Energy Management Systems (HEMS), which are primarily aimed at satisfying user comfort with the available power, minimizing energy consumption, and consequently balancing the supply-demand relationship [10].

There are primarily two methods for load monitoring and identification [11]. One of them is Intrusive Load Monitoring (ILM), which identifies devices with the help of sensors individually installed at each electrical connection. The main basis of this technique is smart plugs, where each time an appliance is turned on, the sensor captures this information and provides it to the consumer through a communication channel [12].

The second method has garnered greater interest within the scientific community due to its economic and privacy advantages during implementation. Non-Intrusive Load Monitoring (NILM) requires only a single measurement point (typically at the main distribution panel) and, with the help of a disaggregation algorithm, it can identify different electrical appliances. NILM is a cost-effective and privacy-preserving method, as it does not require numerous smart plugs or dedicated modules. This technique relies on the fact that each appliance exhibits specific electrical signatures when it is turned on or off, which serve as features for classification algorithms [13].

Based on the above and the opportunities identified previous studies, this work presents the design and implementation of an IoT NILM system, distinguished by the integration of the following characteristics into a cohesive solution:

- Incorporation of a Smart Meter (SM) equipped with a real-time event detection and disaggregation algorithm: The capability of monitoring energy consumption in small time steps allows the NILM system to detect appliance activations,

facilitating accurate load classification.

- Deployment of an unsupervised approach to generate a self-adaptive dataset tailored for end users: Data collection is customized based on user habits without manual intervention, enhancing appliance recognition accuracy and energy management precision.
- Integration of a Deep Learning Algorithm for real-time classification: Multiple deep learning architectures were evaluated to identify the most suitable model for deployment on the web platform, ensuring high accuracy and low latency during appliance recognition.
- Usage of a lightweight IoT NILM framework: This facilitates seamless integration, minimizing hardware requirements, enabling widespread adoption, as well as scaling and cost effectiveness.
- Implementation of the NILM system in a real-world environment: System efficacy validation outside controlled environments, ensuring practical applicability, reliability, and relevance.

1.1 State-of-the-art

Monitoring electrical energy consumption at the device level has been a topic of interest for both the commercial sector and the scientific community since electricity became widely adopted and demand began to match production levels. The concept of Non-Intrusive Load Monitoring began in the 1980s with George W. Hart, who conducted experiments to study residential photovoltaic systems, collecting measurements at 5-second intervals—a data rate significantly higher than the standard 15-minute interval used by utility companies at that time. When analyzing and plotting the collected data, Hart quickly realized that he could "read" the traces visually and identify the behavior of household appliances in the monitored homes. This observation inspired him to formalize the steps for developing a computer program capable of performing a similar analysis to what he had done by sight. It was not until 1992 that he published the article *"Nonintrusive Appliance Load Monitoring"*, where he demonstrated how the energy consumption of individual electrical appliances could be determined through a detailed analysis of the current and voltage of the total load, measured at the interface with the power source [14].

Although the origins of Non-Intrusive Load Monitoring date back to the 1980s, it has gained significant relevance over the past decade, primarily driven by the search for techniques to optimize electrical energy consumption. Additionally, there are technological factors contributing to this resurgence, such as the growing use of smart meters—devices that enable real-time or near real-time monitoring of electrical consumption in buildings, a crucial feature for NILM.

Moreover, the development and implementation of new classification algorithms, artificial intelligence, deep learning, and related technologies have also led to a renewed interest in NILM systems [15]. For instance, Kim et al. [16] proposed a method to identify electrical events for appliance load monitoring based on distinctive electrical characteristics. Their approach detects events by matching extracted feature vectors with entries in an existing appliance database. The analysis focused on power fluctuations and electromagnetic interference noise, as these indicators proved effective for classifying on/off state transitions in various appliance types. To determine the source of each event, the supervised learning algorithm k-Nearest Neighbors (k-NN) was applied alongside an optimization strategy.

The authors of [17] proposed a non-intrusive assistance and guidance system for smart homes, based on the identification of electrical devices, aimed at helping individuals with cognitive impairments remain at home. The core of the system relies on an algorithmic approach designed to recognize erratic behaviors associated with cognitive deficits, and it also provides prompts to guide the user through daily tasks. A novel feature of this system at the time was its ability to deliver cognitive assistance tailored to specific types of cognitive errors, such as step omission, step inversion, perseveration, temporal misalignment, and cognitive overload. To achieve this, the researchers analyzed the load signatures of household electrical appliances—specifically, active power (P), reactive power (Q), and line-to-neutral voltage—to detect errors made by the resident. The system employed a single power analyzer installed on an electrical panel within a real-world smart home prototype equipped with appliances typically used by the patient during their morning routine. Additionally, various multimedia support devices (iPad, screen, speakers, etc.) were utilized. The system was tested using realistic case scenarios modeled after previous clinical trials, demonstrating its potential in terms of accuracy and effectiveness in assisting residents with cognitive impairments in completing their daily activities.

Lin et al. [18] stated that fully developed NILM solutions are still lacking and that further research is needed in this area. Based on this observation, they proposed an intelligent algorithm for appliance identification using 0-1 quadratic programming.

The algorithm is designed to be applied to meters or sensors installed at the main distribution panel of a residential home. To evaluate the proposed algorithm, they built a laboratory-based simulation platform using ten different types of household appliances. The test results showed that the algorithm achieved an identification accuracy of approximately 90%. Similarly, Zhou et al. [19] proposed a method for detecting the on/off states of electrical appliances in home or office settings by measuring active power. They introduced a technical improvement that yields more accurate results using less electrical data. This was achieved by transforming the original load curve, which includes transient changes, into a stepped load curve that better represents steady-state appliance behavior. They then developed a simple, solvable integer programming model to identify appliances and employed a power state classification method to narrow the range of possible devices, enabling a rough analysis of operating states without prior knowledge of the appliances.

Khan et al. [13] developed a hardware setup to measure the electrical parameters of various devices within a system. Using a disaggregation algorithm, the system detects electrical events, which are then processed by a machine learning algorithm to identify appliances. The classification techniques used were k-NN and Random Forests (RF). Each classifier identifies devices by comparing changes in the system's electrical parameters with known signatures stored in a database. Their setup uses an SCT013 current transformer and a 230–9 V voltage transformer for measurements. The machine learning algorithm was implemented in Python and deployed on a Raspberry Pi to enable real-time device identification.

In [20], the authors conducted a study on electrical load identification using fuzzy clustering based on non-intrusive monitoring of electrical signals in non-residential buildings. They proposed a high-resolution machine learning classifier capable of processing data in ten-second windows to detect low-power events and classify load types, regardless of the number of devices connected to the system. Their methodology involves calculating energy consumption from acquired data and identifying events through statistical analysis. These events are then classified using a C-means clustering algorithm based on load type and energy use. Finally, the classified events are placed on a daily timeline with ten-second resolution to visualize load performance throughout the day. This approach was validated using a testbed with resistive, capacitive, and inductive loads, and the experiments were carried out at a healthcare center in the Castilla y León region of Spain.

The state-of-the-art review can also include the work of Zheng et al. [21], which focuses on event-based supervised NILM for nonlinear devices. The study confirmed

that feature extraction based on harmonics exhibits good additive properties, regardless of the practical conditions of the electrical grid. It also found that the Density-Based Spatial Clustering Algorithm with Noise (DBSCAN) outperforms other existing methods for detecting events and adjacent steady states. However, the authors acknowledge that this approach is limited to on/off nonlinear devices and multi-state appliances. While many nonlinear loads exist in buildings, some devices with minimal current distortion—such as vacuum cleaners, air conditioners, and refrigerators—cannot reliably be distinguished using steady-state harmonic feature extraction methods.

Addressing a specific challenge, Shi et al. [12] introduced a novel algorithm called Similar Time Window (STW) for performing NILM using low-resolution data—defined as energy measurements recorded at intervals of 5 minutes or more. Data with shorter intervals are considered high-resolution. Derived from the k-Nearest Neighbors approach, the STW algorithm compares time and frequency domain similarities between the target window and segments from historical data. It then uses instance-based learning to select the most similar windows and estimate device-level energy consumption. Key advantages of the STW algorithm include reduced cost and hardware requirements, improved privacy preservation, and enhanced computational efficiency. To support practical and cost-effective NILM deployment, the study also explores how input/output resolution, time window length, and prediction accuracy are related.

To address some common issues with current smart plugs (SPs)—such as high cost and complex manufacturing—Yu et al. [22] applied NILM techniques to design a functional smart plug capable of identifying connected appliances. By using a relative Euclidean distance algorithm and a vector distance algorithm, the smart plug learns the electrical parameters of various appliances from a database of feature samples. This allows it to automatically recognize the type of device connected and provide users with detailed electrical information, including voltage, current, active power, power factor, frequency, and cumulative energy consumption.

Morais and Castro [23] introduced a novel strategy for NILM. Their method trains a set of auto-associative neural networks, each tuned to the features of a specific electrical device. These networks are arranged in parallel and compete when a new input vector is presented. The one with the closest match identifies the appliance. The system is designed to recognize specific devices using transient power signals from on/off events. The method was evaluated using three public datasets: the REDD (Reference Energy Disaggregation Dataset), UK-DALE (UK Domestic Appliance-Level Electricity), and Tracebase, which contains real residential measurements. Results demonstrated that the technique effectively distinguishes between appliances and that the competitive

architecture achieves good accuracy and F-score performance.

Focusing primarily on load disaggregation, Xiao et al. [24] proposed a load event matching method based on graph theory, built upon an improved Kuhn-Munkres algorithm. Their approach models the problem of matching appliance on/off events and identifying the corresponding loads as a bipartite graph. The enhanced Kuhn-Munkres algorithm is then applied, incorporating both core density probabilities and edge weights to solve the matching matrix, enabling optimal pairing and identification of appliances from a database. Experimental results using the REDD dataset and laboratory data demonstrated strong performance in both event matching and load identification. The authors anticipate expanding their load identification matrix to include more devices and corresponding events, aiming to scale the model for broader applications in the near future.

In [25], a NILM method that incorporates Appliance Usage Patterns (AUPs) to improve device identification and active load forecasting was presented. The authors first identified AUPs for a specific household using a standard spectral decomposition algorithm. These learned patterns were then used in a fuzzy logic system to bias the prior probabilities of appliance activation. AUPs represent the likelihood of a device being active at a given moment, based on recent activity or inactivity and the time of day. As a result, the prior probabilities informed by AUPs enhance the NILM algorithm's accuracy in identifying active loads. The proposed method was successfully tested on two standard datasets containing real household measurements from the U.S. and Germany. Additionally, the authors developed and successfully implemented a residential energy forecasting mechanism capable of predicting total active power demand across multiple households up to five minutes in advance, using the AUP-based approach.

Fang et al. developed a NILM model using device-specific Lightweight Long Short-Term Memory (LSTM) networks, combining supervised and unsupervised learning. Their method performs event detection and sample generation in a fully unsupervised way using a non-parametric Bayesian approach. Each device has its own LSTM, trained on the difference in aggregate load before and after on/off events. These inputs allow each network to predict the current state of its corresponding appliance. By assigning a separate LSTM per device, the method reduces learning complexity and improves detection of overlapping switching events. Controlled experiments showed that this approach outperformed several existing methods [26]. They also conducted ablation studies to assess each module's contribution and evaluated the effect of different synthetic data distributions.

Another real-time NILM application was presented by Mihailescu et al. [27], who designed a low-cost hardware solution for appliance recognition using point-to-point IoT communication. They conducted a comparative analysis to identify electrical signals from various devices. First, they assessed different machine learning methods using publicly available datasets, which typically include several months of low-frequency data. Then, they evaluated these techniques in their specific real-time scenario, aiming for rapid device identification using a limited training dataset. This focus on data-efficient learning is critical, as high variability in appliance usage patterns often requires collecting device-specific data to achieve accurate recognition—a process that is costly and impractical at scale. In addition to their end-to-end IoT solution, the authors analyzed its computational demands in terms of cost and real-time performance, both essential factors for low-power systems..

Akarslan and Dogan [28] introduced a device identification method that relies solely on current waveforms for feature extraction. First, they preprocess the signal to isolate one period, then compute the Fast Fourier Transform (FFT). The real and imaginary parts of the FFT are evaluated separately, and statistical features such as maximum, minimum, and standard deviation are extracted from each. These features define thresholds for different devices, forming a rule table used for identification. The method can recognize both individual appliances and their combinations. Results show an identification accuracy above 98%, with a 4.7% improvement achieved by separately analyzing the real and imaginary FFT components.

Another noteworthy study is presented in [29], where the authors propose an on-site, self-adaptive non-intrusive load monitoring method. Their goal was to enable fully automated operation and accurate NILM results in real-world environments without requiring prior information. To achieve this, they developed a self-adaptive signal database during actual operation, allowing for accurate load data collection that significantly improves identification. Because accurate classification depends on having enough information, they used high-frequency data acquisition to preserve full waveforms and mixed load signals. Their approach follows three main steps: (1) decomposing the mixed signal into separate, unlabeled load components; (2) identifying load types based on inherent features and statistical characteristics; and (3) classifying them using a Bayesian model trained on the evolving database.

In a related study, [30] addressed near real-time load identification from low-frequency energy data in office settings. They first detected active load periods from outlets, then applied a dynamic time window strategy for feature extraction. These features were fed into the Bagging algorithm with a minimum 5-minute dynamic window

resulting in 93% accuracy.

In [31], Fan et al. provided a load disaggregation model built on a multi-objective function that captures both macro- and micro-level appliance features, offering a comprehensive representation of device behavior. Their approach uses five objective functions based on apparent, active, and reactive power, current, and harmonics to recognize various appliances. Treating NILM as an optimization problem, they apply the Moth-Flame Optimization (MFO) algorithm, highlighting its ability to integrate multiple functionalities and diverse signal features—capabilities rarely seen in prior NILM approaches. To handle the fact that most appliances operate in multiple states, the model incorporates a Factorial Hidden Markov Model (FHMM) to define allowable appliance modes over each second. The effectiveness of their method is supported by experimental results and comparisons with existing state-of-the-art techniques. They also tested different combinations of appliance features and found that performance improves with feature diversity—achieving at least 20% higher accuracy than previously reported methods.

Chen et al. [32] argue that most existing load identification algorithms rely heavily on steady-state or transient electrical features, limiting their performance due to constrained feature selection and analysis patterns. To address this, they propose a deep neural network approach for NILM using a testbed configured with various household appliances to collect real-time energy usage data in a typical Chinese home. The collected data were preprocessed and fed into a hybrid Convolutional Neural Network–Long Short-Term Memory (CNN-LSTM) model for training and feature extraction. Experimental results showed that the CNN-LSTM outperformed traditional methods such as k-NN, SVM, standalone CNN, and LSTM models, achieving an average recognition accuracy of 99% across different types of appliances in China's typical grid.

In contrast, Himeur et al. [33] developed a NILM method based on a novel 2D local energy histogram descriptor combined with an enhanced k-NN classifier. Their technique captures robust device-level features from aggregated power signals using short local histograms, improving classification and reducing computation time. This approach yielded impressive accuracy rates—99.65% on the GREEND dataset, 98.51% on UK-DALE, and over 96% on WHITED and PLAID—demonstrating the effectiveness of 2D descriptors in precise appliance identification and offering new insights for NILM research.

Ahammed et al. [34] developed a real-time IoT-based NILM system using low-frequency data to address the lack of smart meters in Bangladeshi homes. They built a custom acquisition device measuring RMS voltage/current, active power, and power

factor at 1 Hz. The best-performing multi-label classifier was deployed using Firebase for real-time data exchange and a GPRS module for wireless transmission without Wi-Fi. Tested in a real home, the system maintained over 94% accuracy despite voltage fluctuations, supporting its potential for efficient energy management.

Similarly, Franco et al. [35] proposed an IoT-based system for load monitoring and activity recognition. Unlike traditional systems relying on tagging, they applied machine learning models—FFNN, LSTM, and SVM—for device identification. FFNN and LSTM achieved precision above 0.9, outperforming SVM. The best model was then used to detect daily activities based on device usage patterns. They also conducted a sensitivity analysis to assess how group size impacts classification accuracy.

Yin et al. [36] focused on identifying unknown residential loads and updating existing classification models. Their approach also aids in detecting abnormal current changes and potential electrical faults. They highlight three main contributions: (1) a hybrid Siamese neural network and isolation forest model that extracts low-dimensional features, filters outliers, and identifies unknown devices using convex hull overlap rates; (2) a shared-parameter transfer learning method to quickly update classification models; and (3) validation using both public and proprietary datasets. Their results show that the presented model effectively detects unknown loads and updates itself with high accuracy.

Li et al. [37] addressed the inconsistency in NILM performance caused by relying on single features. To improve generalization and accuracy, they proposed a method that deeply fuses multiple features and classifier outputs using Dempster-Shafer (D-S) evidence theory. They extracted power, current harmonic, and voltage-current trajectory features, then used k-NN, random forest, and CNN classifiers independently. The outputs were combined using D-S theory to improve overall prediction. Compared to standalone models, their approach improved classification accuracy and F1 score by 5.23% and 3.03%, respectively, demonstrating that decision-level fusion yields better results than using individual classifiers or existing methods.

In a similar line of research, Athanasiadis et al. [38] presented a real-time, multi-class NILM system. To begin with, it uses a robust adaptive thresholding event detection algorithm, followed by a deep CNN and k-NN classifier to identify various devices in real time. What sets this system apart is its ability to detect appliance activation as it happens while remaining lightweight enough for low-cost chips or smart meters—making large-scale NILM deployment more practical. Importantly, the system does not require device-specific training. Instead, it transforms transient responses into 30-dimensional vectors that naturally cluster by appliance, eliminating the need

for large, high-frequency datasets and costly retraining. For validation, they used the BLUED dataset for event detection and tested the k-NN classifier with private data, showing strong real-world performance. Finally, extensive comparisons demonstrate that their system outperforms most state-of-the-art real-time methods.

Most NILM systems rely on supervised learning, which is simpler to design but time-consuming to implement. To address this, Zhou et al. [39] proposed an unsupervised NILM algorithm using spiking neural networks (SNNs), which require no labeled data. SNNs are ideal for low-cost deep learning hardware, as each neuron can be implemented with basic RC circuits. The authors also introduced a distributed computing setup, splitting the SNN between smart plugs and local servers—allowing high-frequency sampling without the need for powerful communication hardware. Tests using the PLAID dataset showed 83% accuracy, suggesting their unsupervised method performs comparably to traditional deep learning-based NILM approaches.

Unlike traditional NILM applications, Lin et al. [40] proposed a smart home energy management system using an IoT-based NILM approach grounded in multi-objective evolutionary computation, entirely training-free for demand-side management in smart grids. Different from AI-based methods such as artificial or deep neural networks, which require training, retraining for new appliances, and hyperparameter tuning, their model relies on combinatorial optimization using genetic algorithms. Their NILM strategy treats appliance identification as a pattern-matching problem, comparing potential appliance profiles against measured aggregate energy consumption using metaheuristics to minimize error. The system was deployed in a real home environment, and experimental results confirmed its feasibility.

Liu et al. [41] introduced a load identification method that combines LSTM networks with Affinity Propagation (AP) clustering. Their approach begins by preprocessing raw energy consumption data using the AP algorithm to group large volumes of samples into multiple clusters. To handle continuously variable loads, they discretize device power profiles into representative sets. These processed data are then input into an LSTM-based classification model, which outputs the most likely operating states and power levels of appliances, accounting for potential prediction errors. When comparing results from the standalone LSTM model versus the combined LSTM-AP approach, they found that the latter reduced deviation from real load data to between 2% and 7%—a significant improvement over using LSTM alone.

Similarly, inspired by the limitations of training-heavy classification models, Yu et al. [42] proposed a lightweight alternative using a Siamese neural network—a few-shot learning approach that performs well even with limited training data. Their architecture

centers on a convolutional neural network, commonly used in transfer learning due to its generalizability. This setup computes the similarity between test samples and a reference feature library to either recognize existing devices or detect new ones. Input features include voltage and current waveforms along with active power. Their method greatly reduces the reliance on extensive training datasets, adding flexibility to NILM implementations. The model's effectiveness was validated using the PLAID and COOLL datasets.

Recent studies have advanced NILM performance through diverse approaches. Liu et al. [43] used LSTM networks with data augmentation and clustering to better recognize appliance usage patterns. Winkler et al. in [44] proposed a low-cost, single-device smart meter based on Raspberry Pi and YoMoPie Monitor, providing local real-time feedback while addressing privacy concerns and achieving promising disaggregation performance. The authors of [45] introduced the Next-Generation Smart Grid Meter (NGSM), an embedded Linux-based system that monitors household energy, communicates via a smart grid, and applies deep learning for appliance classification, achieving 93.75% accuracy with three devices operating simultaneously. These works illustrate the trend toward combining machine learning, compact hardware, and on-edge processing to enhance NILM effectiveness.

Finally, it's worth highlighting efforts focused on reviewing the state of the art in NILM research. One of the earliest examples is by Najmeddine et al. [46], who in 2008 provided an overview of the most prominent NILM methods at the time, outlining their strengths and weaknesses in the search for the most effective approach to load identification.

While most of the NILM research relies on public or private datasets for validation, the performance of each technique varies depending on the dataset's characteristics. To address this, Khurram et al. [15] conducted a comprehensive review of 42 NILM datasets, presenting comparative tables to highlight their features. The paper also outlines current limitations and strengths of these datasets, providing insights into existing challenges and future research directions. Their goal is to help researchers evaluate algorithm performance and guide the development of new datasets in the NILM community.

In 2016, Abubakar et al. [9] reviewed recent trends in energy management to guide researchers in the field. They analyzed and categorized various HEMS strategies using both ILM and NILM techniques. Their findings highlighted several challenges: the need for more accurate load recognition, systems capable of identifying a wide range of appliances, greater efforts to integrate NILM into appliance-level energy management,

and the importance of fostering energy awareness among consumers across homes, offices, and industries.

Later, in 2021, Khurram et al. [15] conducted a comprehensive review of the most widely used datasets for validating NILM techniques, identifying REDD, Smart*, UK-DALE, BLUED, and AMPds as the top five. In 2022, Angelis et al. [47] examined various machine learning algorithms applied to NILM, discussed recent advancements, and highlighted ongoing limitations and research challenges.

That same year, Yan et al. [48] explored the challenges and improvement opportunities across each stage of a typical NILM pipeline, with a focus on enabling real-time applications. Also in 2022, Dash et al. published a systematic review of the NILM field, structured around two key research questions: (1) How has the research community approached and evolved NILM solutions? and (2) How do different NILM approaches compare in terms of key attributes? Their work helped outline critical challenges that still lie ahead in advancing NILM technologies.

Latest reviews have provided comprehensive insights into the current state and challenges of NILM. Rafiq [49] summarized advanced deep learning-based methods in residential contexts, highlighting key limitations and research gaps. Luo [50] emphasized the difficulties in scaling NILM from laboratory settings to real-world applications, including diverse consumption patterns, complex disturbances, decentralized data, and limited computing resources, and discussed advances in robustness, adaptability, collaboration, and deployability. Nexus [51] extended the discussion to industrial settings, noting that online NILM can enable appliance-level monitoring, optimize load scheduling, reduce energy consumption, and improve operational efficiency, while also detailing relevant hardware implementations. Together, these reviews map the evolution of NILM research and outline the practical challenges that remain for widespread adoption.

Using the most representative and complete works found in the state of the art, Table 1.1 was created to compare their main features. The analysis focuses on six key parameters: the use of low-frequency parameters, real-time event detection, automatic tailored dataset generation, unsupervised learning, IoT platform integration, and validation in real environments. This highlights where the present work stands with respect to existing solutions.

Table 1.1: Comparison among existing NILM works.

Reference	Low frequency parameters	Real-time event detection	Automatic tailored dataset	Unsupervised Approach	IoT Platform	Deployment in a real environment
[13]	✓	✓	×	×	✓	✓
[52]	×	✓	×	×	×	×
[23]	✓	×	×	×	×	×
[39]	×	×	×	✓	×	×
[53]	×	×	×	×	×	✓
[42]	×	×	×	×	×	×
[26]	✓	×	✓	✓	×	×
[34]	✓	✓	×	×	✓	✓
[29]	×	✓	✓	✓	✓	✓
[54]	×	✓	×	×	×	×
Present work	✓	✓	✓	✓	✓	✓

1.2 Problem Statement and Justification

Global electricity consumption continues to rise, driven largely by population growth and the rapid urbanization of previously underserved regions [55]. This increasing demand is not only due to expanded access to electricity, but also to the growing use of energy-intensive appliances—such as air conditioners and heaters—which are now common even in areas where they were once rare.

The early 21st century has also seen a surge in the use of portable electronic devices like smartphones, laptops, and smartwatches, all of which require regular charging and thus contribute to overall electricity demand [56, 3]. According to data from Our World in Data, global electricity generation increased by approximately 32% between 2010 and 2025 [57]. This upward trend is expected to continue, particularly with the widespread adoption of energy-intensive technologies such as electric vehicles, cryptocurrencies, blockchain, and advanced generative language models—all of which place additional strain on power systems.

Why is this a problem? From a production standpoint, the primary concern is that 59% of global electricity was still generated from fossil fuels as of 2024 [58]. These non-renewable resources not only deplete over time but also contribute significantly to global pollution and climate change. Addressing rising energy demand is therefore critical—not only for sustainability, but for the health of the planet.

Considering the current state of distribution infrastructure alongside the sharp rise

in energy demand discussed earlier, it is clear that this issue has intensified over the past two decades. Many power plants and grid systems were not designed to handle these modern consumption levels, leading to issues like reduced supply during peak hours and, in some cases, extended outages [3]. This growing demand presents several challenges, which can be addressed by improving energy efficiency. One promising solution is the development of smart grids—systems that balance supply and demand through intelligent management and modern communication technologies [3, 4, 5, 6].

At the core of smart grids are smart meters, which enable two-way communication between consumers and utility providers—a key feature of smart grid functionality. These devices do more than record energy usage; they also transmit electrical parameters in real-time or at scheduled intervals, making energy consumption more transparent and manageable.

In recent years, research has focused on Home Energy Management Systems (HEMS), which aim to reduce energy waste, minimize blackouts, and benefit both consumers and utility companies [59]. Smart meters play a crucial role in HEMS by providing real-time data that enables analysis, control, and optimization of household energy use. HEMS can be implemented using intrusive load monitoring, which requires a meter on each device, or non-intrusive load monitoring, which disaggregates household-level energy data from a single smart meter to determine individual appliance usage. Each approach has its own pros and cons, which are discussed in later sections. What is important here is that both aim to identify and classify household appliances based on their energy usage.

Appliance-level consumption data offers several benefits. It empowers consumers with detailed insights into their energy habits, enabling behavioral changes that can lead to savings. It also supports monitoring the health of devices, as malfunctioning appliances often consume more power, generate electrical harmonics, or even pose safety hazards such as fire risk [60].

Beyond individual benefits, appliance-level data improves understanding of demand profiles for small-scale consumers. This insight supports accurate forecasting of peak, average, and shifting demand trends [23, 61, 32]. It also facilitates the better integration of renewable energy sources in microgrids by helping to maintain financial and operational balance.

Renewables like wind and solar introduce supply-side uncertainty due to weather variability. While solutions like wind speed or solar radiation forecasting and storage optimization are well-studied, less attention has been given to demand-side variability. Understanding device-level demand behavior can reduce load uncertainty and help

optimize battery sizing or intelligently schedule appliance use within microgrids [21].

While the benefits of load monitoring are clear, both NILM and ILM face important challenges. ILM relies on installing a smart meter at each appliance, which makes its feasibility heavily dependent on hardware costs, system design, and available budget. The practicality of ILM is thus limited by the cost-functionality tradeoff in its implementation.

NILM, on the other hand, presents a different set of issues. Although the concept originated in the 1980s, it has gained renewed interest thanks to advances in artificial intelligence. NILM depends on algorithms for energy disaggregation and device classification using a single point of measurement—typically the total energy consumption of a building. However, one of NILM's core limitations is the lack of algorithms to reliably identify individual loads, making this an active and open area of research [62, 23, 61, 18].

As noted in [9], NILM research faces several persistent gaps. Many studies prioritize simulation environments and public database validation over real-world deployments, limiting their practical applicability. Feature extraction, a crucial component in machine learning-based NILM, also lacks standardization, with no widely accepted set of electrical characteristics for appliance-level identification.

In addition, most NILM models are applied offline using historical data, while real-time load identification remains relatively unexplored. Another major limitation is the scarce integration of NILM techniques within IoT frameworks, which restricts their adaptability to modern smart home infrastructures. Without real-time capabilities and seamless IoT integration, NILM remains constrained to controlled research environments, falling short of the responsiveness and interoperability required for deployment in dynamic, connected households.

Particularly underrepresented in the current literature is the automatic creation of tailored appliance datasets—an essential capability for developing self-adaptive systems that can adjust to new environments without manual training. This area remains largely unaddressed, despite its potential to significantly enhance the scalability and generalization of NILM solutions. Addressing scalability is critical for real-world adoption, as practical NILM systems must be capable of handling diverse residential environments, varying electrical infrastructures, and the continuous introduction of new appliances. Without scalable approaches that combine real-time operation, IoT integration, and automatic dataset generation, NILM will remain limited to small-scale research prototypes rather than evolving into widely deployable solutions for large numbers of households and future smart grid applications.

Based on the identified limitations, this work presents the design and implementation of an automatic and self-adaptive NILM system using unsupervised learning and an IoT platform. The proposed approach applies NILM by leveraging a unique smart meter to capture key electrical parameters from the main distribution panel, integrating event detection and disaggregation algorithms with a real-time neural network model deployed through a web application.

1.3 Research Hypothesis

An IoT-based NILM system that integrates real-time event detection within an embedded device, combined with unsupervised learning and deep learning techniques, can enable scalable and self-adaptive appliance-level classification, capable of adjusting to diverse residential environments without relying on public extensive datasets, high-frequency electrical parameters, or manual data collection.

1.4 Research Objectives

1.4.1 General Objective

To develop an IoT-based Non-Intrusive Load Monitoring system that employs real-time event detection and unsupervised learning techniques to automatically generate a self-adaptive appliance dataset for accurate load classification in real residential environments.

1.4.2 Specific Objectives

- To build a smart meter that integrates a real-time event detection and disaggregation algorithm capable of accurately identifying at least 95% of appliance switching events from aggregated load signals.
- To apply unsupervised learning techniques to automatically generate a dataset comprising data from a minimum of five different appliances.
- To evaluate at least three deep learning models for time series classification, selecting the model that balances computational efficiency and achieves over 95% classification accuracy.

- To develop a web application that integrates a classification algorithm to recognize events with a response time of less than 3 seconds.
- To validate the scalability and robustness of the NILM system by deploying it in no fewer than two real residential environments, ensuring accuracy exceeds 95% across all cases.

1.5 Methodology

Given the multidisciplinary nature of this project, it was essential to follow a well-structured development process. Although the proposed system operates using data acquired from a smart meter, it is important to note that the smart meter hardware has already been fully developed prior to this work, following an IoT-based embedded systems approach. Therefore, the structure of this thesis document lies in the design and implementation of the NILM system. To guide this development, the waterfall methodology was selected, a sequential design model commonly used in software engineering, where progress flows linearly through clearly defined stages, as illustrated in Figure 1.1 and described below [63].

1. **Requirements Analysis:** Gather all relevant information and define the project's specific needs. This includes identifying deliverables, setting clear expectations, and assigning roles so each team member understands their responsibilities.
2. **Design:** Define the technical specifications of the system, such as the programming language and hardware requirements. This phase outlines the overall system architecture and details how each component functions and interacts with others.

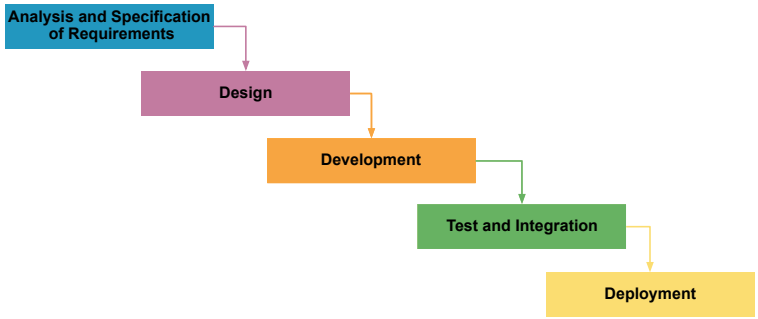


Figure 1.1: Waterfall methodology steps.

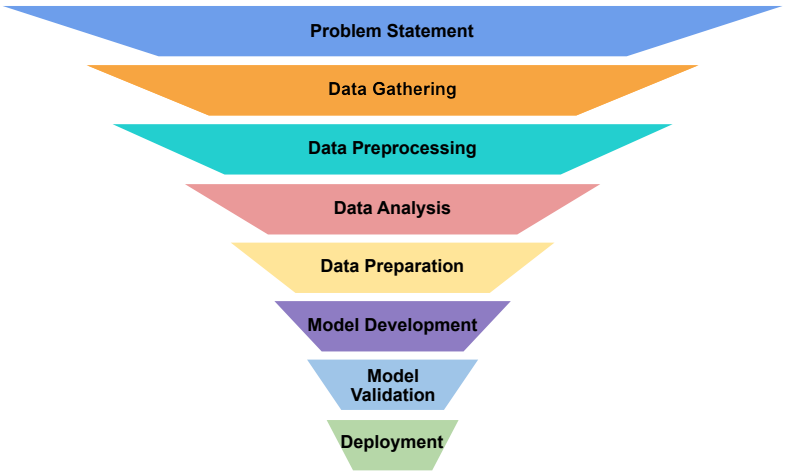


Figure 1.2: Workflow for Machine Learning Development.

3. **Development:** Implement the planned design by writing the necessary code and algorithms. This stage involves iterative prototyping, testing, and debugging to meet user requirements.

4. **Testing and Integration:** Combine all developed components into a complete system and verify that everything works as expected. Errors are identified and resolved before the system is released.

5. **Deployment:** Finalize the project by making the software available to its intended users, ensuring it is ready for real-world operation.

Focusing on the development phase, this is where the necessary code is implemented to carry out the project tasks. In this case, since the core of the project relies on machine learning algorithms, a specialized workflow was adopted. The steps are shown in Figure 1.2 and described below:

1. **Problem Definition:** The problem should be framed within one of the standard machine learning categories—classification, prediction, clustering, or decision-making. This helps determine the type and amount of data needed, as well as the appropriate learning model. A poor abstraction can lead to an unsuitable model and underwhelming performance.

2. **Data Collection:** This step involves gathering the necessary data based on the problem type. At this stage, a relevant dataset is either created or selected for use.

3. **Data Preprocessing:** Before extracting features, raw data must be cleaned and

prepared. This can include tasks such as filtering, discretization, and imputing missing values.

4. **Data Analysis:** Depending on the problem, understanding the general data trends is crucial. Statistical distributions and data visualizations help reveal correlations and patterns that guide model design.

5. **Data Preparation:** Even after preprocessing and analysis, additional steps like transformation, normalization, or standardization may be required—especially when data features operate on different scales.

6. **Model Building:** An appropriate learning algorithm is selected based on the problem type and dataset size. The dataset is then split into training and testing sets for model development.

7. **Model Validation:** This critical phase tests the model's performance. Cross-validation is often used to assess accuracy and detect overfitting. Insights from this step help refine the model—by increasing data volume or simplifying the model architecture if needed.

8. **Deployment:** When deploying the model in a real-world setting, practical considerations such as computational power, energy consumption, and response time must be addressed. Balancing accuracy with system efficiency is key to effective implementation.

Thanks to the rapid advancement of NILM research, the scientific community has largely adopted a standard methodology structured around four key stages: data acquisition, event detection, feature extraction, and load identification (see Figure 1.3) [51, 64].

1. **Data Acquisition:** Electrical signals—such as current, voltage, and power—are collected from a smart meter. Raw data is then filtered to remove noise.

2. **Event Detection:** An event refers to a device changing state over a short period. These events are typically detected by identifying variations in power or current that exceed a predefined threshold.

3. **Feature Extraction:** Device characteristics are extracted from the electrical signals using techniques like the Fourier Transform. These features provide a numerical representation of each load, usually formatted as vectors whose dimensions depend on the number of extracted attributes.

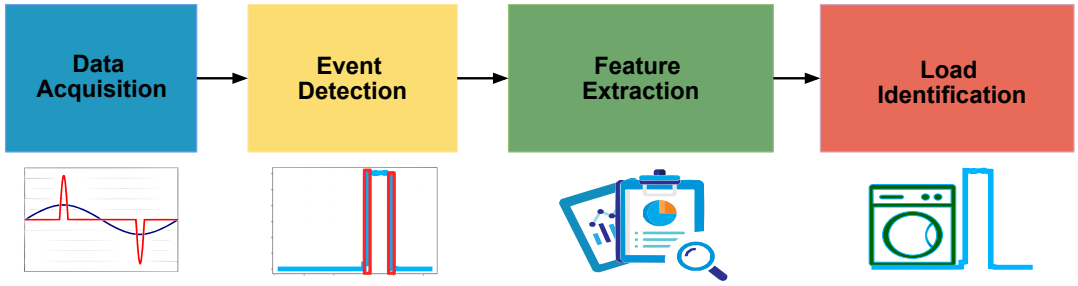


Figure 1.3: Methodology for NILM projects.

4. **Load Identification:** This step applies supervised or unsupervised learning algorithms to the extracted features to classify and identify which appliance corresponds to each detected signal.

Chapter 2

Theoretical Principles

The development of an automatic and self-adaptive NILM system relies on a solid theoretical foundation that integrates concepts from electrical engineering, signal processing, machine learning, and IoT. This chapter presents the essential principles underlying the proposed system, beginning with the fundamentals of load monitoring and the classification of steady-state and transient features in electrical signals. The analysis of electrical parameters at both high- and low-frequency sampling rates is then introduced, covering raw current signals, V–I trajectories, harmonic components, and power-related quantities such as active, reactive, and apparent power. Artificial intelligence techniques are also reviewed, with an emphasis on machine learning and deep learning models commonly applied in NILM research, together with clustering methods, convolutional architectures, and recurrent network models. The chapter concludes by examining the role of IoT in enabling data acquisition, processing, and communication through layered architectures, thereby establishing the theoretical basis for the design and implementation of the proposed NILM system.

2.1 Load Monitoring Concept

Load monitoring is the process that involves the acquisition and identification of load measurements within a power system. Collecting load measurements can provide insights into energy usage and the operational status of appliances, enhancing the understanding of how individual loads contribute to the overall energy consumption within the system [9]. The purpose of load monitoring is to enable energy conservation by adopting efficiency-oriented strategies, such as employing energy-efficient devices, optimizing appliance usage timing, and reducing unnecessary power consumption activities [65]. Without a proper monitoring system, it can be difficult to determine

the operational status of appliances. The load monitoring systems can be divided into intrusive and non-intrusive.

2.1.1 Intrusive Load Monitoring

This approach is about installing measuring devices at each specific load of interest, that is why it receives the term intrusive. For instance, if 'n' appliances within a particular area of the power grid will be monitored, then 'n' meters will be required. This extensive instrumentation makes ILM relatively expensive, complex to install, and challenging to maintain. However, it offers the advantage of providing highly accurate measurements (depending on the quality of the meters) and nearly instantaneous load identification [60].

The type of ILM implementation depends on how many appliances share one set of sensors. A. Ridi et al. [66] divided the ILM into three categories: the sub-metering system, where a zone of appliances has one meter; the smart plug-in, where a plug connects several appliances to one meter; and the smart appliances, where each appliance has its own meter or a built-in meter.

A. Ridi et al. also proposed that the ILM framework comprises three steps:

1. **Single Appliance Detection:** Involves utilizing sensor technology, such as smart plugs, to identify when appliances are switched on or off and to determine their specific locations.

2. **Middleware Phase:** This phase involves software that interprets appliance statuses derived from the detection process.

3. **Appliance Status Phase:** This final stage of the ILM provides real-time appliance statuses, facilitating intervention, control, and monitoring.

2.1.2 Non-intrusive Load Monitoring

As mentioned before, intrusive load monitoring involves the use of additional hardware to track the usage of appliances. While this approach is accurate, it faces challenges with user acceptance due to the substantial capital investment required and the need for individual installations per appliance within the ILM system [60]. To address the previous challenges, NILM emerged. Also known as *energy* or *load disaggregation*, this technique aims to distinguish the operational status (ON/OFF) and the exact power usage of individual electrical loads. It achieves this by considering as input only the

aggregated consumption of these loads. A graphical representation of NILM can be seen in Figure 2.1.

2.1.3 Real-time Requirements in NILM Systems

A real-time system is characterized not only by the logical correctness of its computations, but also by the temporal constraints under which those computations are delivered. In other words, the system behavior depends both on the *result* and on the *time instant* at which the result is produced. Such a system is always embedded in a larger real-time environment that evolves with physical time, and it must react to external stimuli within deadlines dictated by that environment [67].

In the context of NILM applied to residential environments, real-time processing ensures that the smart meter can acquire electrical signals, process them, and detect appliance-level events in a timely manner. The usefulness of this analysis is directly linked to whether results (e.g., appliance state transitions) are available when they are needed by the user or the home energy management system.

The instant by which a computation must be completed is called a *deadline*. Depending on the criticality of missing a deadline, real-time systems are categorized as follows:

- **Hard real-time systems:** Missing a deadline leads to unacceptable consequences. Although most residential NILM applications do not fall into this category, there are specific cases where strict timing may be required. For example, a NILM system integrated with a home battery management unit or with an automatic load-shedding mechanism must identify high-power appliance

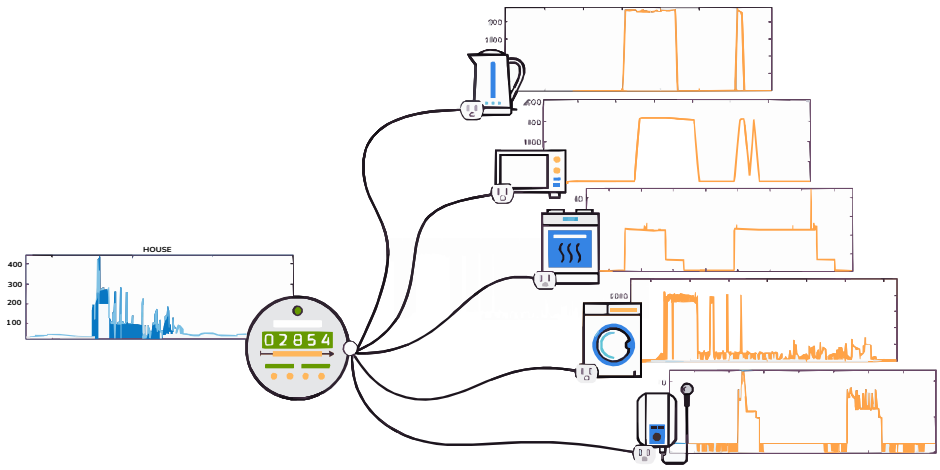


Figure 2.1: Representation of a NILM system.

activations within milliseconds to immediately prevent overload or protect the installation.

- **Soft real-time systems:** Deadlines are important but not absolutely critical. This is the most common category for residential NILM. For example, if a system reports that the washing machine has started a few seconds late, the only impact is a minor reduction in responsiveness for applications such as energy dashboards, demand-response participation, or home automation routines. No safety or functional failure occurs.
- **On-line real-time systems:** These systems tolerate more relaxed deadlines, and results remain useful even when delivered with some delay. An example in a home environment is the aggregation of appliance-level consumption data for daily or weekly reports. Here, delays of several seconds or even minutes are acceptable, since the focus is on trend analysis and long-term feedback rather than immediate action.

This classification highlights that residential NILM typically operates under soft or on-line real-time requirements, but in some cases, especially when integrated with protective or automation mechanisms, it may approach hard real-time constraints. Designing NILM for residential IoT environments thus requires balancing latency, accuracy, and computational resources to meet the appropriate category of real-time demand.

2.2 Analysis of Electrical Parameters in NILM

A fundamental part of non-intrusive load monitoring systems are the electrical parameters to be used to feed event detection and load classification algorithms. Load signature is the specific pattern an appliance produces when it is in working mode. Every electrical appliance has its unique characteristics that define its electrical behavior [47]. Despite various suggestions in literature regarding the features to be employed, researchers have yet to agree on a standard set of electrical parameters that can guarantee high accuracy in the NILM process [60].

The electrical parameters that can describe a load can be divided into steady state or transient features.

2.2.1 Steady State Features

Steady state features are obtained from appliances when they operate in a consistent state. In the context of NILM studies, these features can often be captured using smart meter data with low sampling rates. The electrical measurement most frequently utilized for this purpose is the active power and its temporal fluctuations. The most commonly employed electrical measurement for this purpose is the active power and its temporal variations. Appliances with purely resistive loads, lacking capacitive or inductive elements, can be recognized by utilizing active power as a sole feature. However, relying solely on active power for load classification can result in misidentification due to the possibility of multiple loads overlapping each other. Nowadays, the majority of residential and industrial loads work with phase shifts between current and voltage signals. This indicates that they either generate or consume reactive power. Therefore, incorporating reactive power as a feature has been shown to enhance the predictive accuracy of models [68].

In steady state, power is not the only feature that can be analyzed. For example, [69] highlighted the use of Voltage-Current (V-I) trajectories in steady state conditions. The parameters they focused on were area, slope, curvature, asymmetry, overshoot and trajectory centroid.

Up to now the features mentioned earlier are extracted in the time domain. However, some studies also use signal transformation from the time domain to the frequency domain for extracting steady state characteristics. One example is the application of Fourier analysis to detect steady-state current harmonics.

One widely used approach for examining high frequency signals is to transform them into either the frequency or wavelet domain. Short-Time Fourier transform (STFT) and fast Fourier transform (FFT) are commonly employed techniques to extract current harmonics. This can be helpful in identifying transient load signatures.

2.2.2 Transient State Features

The transient state refers to the period of time when an electrical load is changing from one steady state to another. During this time, the electrical parameters of the system, such as voltage and current, are changing from their initial values to their final values.

Transient state characteristics are obtained by capturing data with a high sampling rate that enables identification of the appliances' transient signatures during operation, as illustrated in Figure 2.2. Simultaneously operating appliances can be a challenge when utilizing steady state electrical parameters in algorithms. However, transient

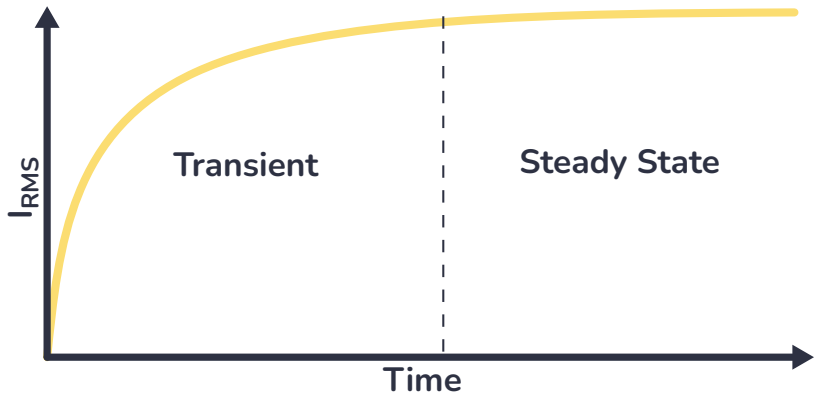


Figure 2.2: Example of transient state and steady state.

features offer a solution as they provide unique information for each appliance and tend to exhibit less overlapping in transient signatures. Not everything is a positive characteristic of transient state features. Although capturing data at high sampling rates can reveal transient features, it may also cause difficulties as some hardware equipment may not be able to process signals at such rates. Additionally, data storage and computational issues may arise from the large volume of data required for training AI models.

2.2.3 Sampling Rate

When working with non-intrusive load monitoring, it is important to understand the terminology used and its relationship to the data being analyzed. In electric meters, voltage and current signals are sampled internally at frequencies that exceed the base frequency of the AC supply. The meter may output the raw data or provide averaged values such as RMS voltage and current, active, reactive and apparent power, or total harmonic distortion (THD) at lower frequencies [70]. In the NILM literature, the following key terms are commonly used to describe the sampling characteristics of the data:

- **Low-frequency** methods: These refer to approaches that analyze features or measurements derived from data sampled at rates lower than twice the AC base frequency signal, or at rates comparable to it (e.g., 1–120 Hz). These typically include statistical values such as RMS or power readings.
- **High-frequency** methods: These refer to techniques that utilize raw voltage and current signals sampled at much higher rates (e.g., several kHz), capturing transient events and waveform details that can be useful for identifying individual appliances.

2.2.4 Classification of Loads

Appliances can be classified into different categories based on how they function over time, including:

- **ON/OFF state:** These appliances are characterized by having only two possible operational states. Examples of such appliances include lamps, toasters, boilers, and other resistive loads.
- **Finite State Machines (FSM):** Devices in this category have a finite number of operating states. For example, washing machines, stove burners, or dishwashers. These appliances typically exhibit a repeating pattern or alterations in their operational states, which can cause difficulties when algorithms try to identify them.
- **Continuously Variable Devices (CVD):** These appliances exhibit a range of power consumption levels, rather than a fixed number of operating states. Examples of CVD appliances include dimmers lights or electric stovetops with variable temperature settings. Because CVD appliances do not have a set number of operating states, they can be difficult to classify.
- **Permanent consumer devices (PCD):** Devices that are constantly plugged in and connected to a power source, such as wall outlets or power strips. Examples of PCDs include refrigerators, televisions, computers, and home entertainment systems. These devices typically consume energy even when they are not actively in use, such as when they are in standby mode or have a "sleep" function.

According to [71] appliances can also be classified depending on their load nature, for example:

- **Resistive Loads**: These appliances use resistive heating elements to convert electrical energy into heat or light. They have a simple design consisting of a heating element made of a resistive material, usually a wire, which heats up when an electric current passes through it. Incandescent light bulbs, toasters, electric heaters, or electric stoves are examples of resistive loads.
- **Reactive predominant loads:** These devices use inductive or capacitive loads to operate, which means that the current flowing through them lags or leads the voltage. This can cause an increase in reactive power and result in higher energy consumption, as well as potential issues with voltage stability and power quality.

Examples of reactive loads include freezers, washing machines, electric motors, or fluorescent lamps.

- **Loads with power factor correction circuit (PFC):** The IEC Standard 61000-3-2 limits harmonic current level for all the loads with power above 75 Watts. So appliances that have a PFC circuit are designed to correct the power factor by reducing the amount of reactive power that the appliance draws from the electrical system. Personal computers over 75 W, projectors, LCD TVs, LED TVs (working in the "high quality mode"), Plasma TVs, home theaters and game consoles, all belong to this category.

- **Loads with no power factor correction circuit:** Electronic loads in this category do not utilize power factor correction techniques. Among them are cell phone chargers, portable DVD players, adapters for portable printers, scanners, and many others.

- **Linear loads:** This category of appliances comprises low-power devices that utilize linear DC power supplies with a small transformer at the front-end. Examples of representative loads in this category include battery chargers, paper punchers, and staplers.

- **Phase angle controlled loads:** These are devices that use thyristors to control their power. For example, light dimmers.

- **Complex structure:** This family of appliances typically has high power requirements and incorporates multiple electrical subsystems. Examples include microwave ovens and laser printers.

2.3 Electrical Parameters at High Frequency Sampling Rate

As mentioned above, electrical parameters at high frequency sampling rate means working with raw voltage and current signals. In this case, meters provide data points at much higher rates than the fundamental frequency. By doing so, algorithms capture more detailed and precise information about the electrical waveforms, enabling a finer-grained analysis of the behavior and characteristics of individual loads within the power system.

2.3.1 Raw Current Signal

The instantaneous current (i) represents the flow of electric charge through a circuit and carries valuable insights about the appliances connected to that circuit. You can see a waveform of alternating current (AC) in Figure 2.3.

Appliances generate unique electrical signatures due to their internal components. These signatures are reflected in the current they draw from the power supply. The switching operations within devices produce rapid changes in the current flow, manifesting as high-frequency components in the current signal. For instance, electronic devices like computers or TVs often exhibit abrupt variations in current consumption as they power ON/OFF, or switch between different modes. These transitions create distinctive patterns in the current signal.

In terms of parameters used for feeding classification algorithms, the current signal falls within the high-frequency category. To capture the high-frequency characteristics inherent in the current signal, the sampling frequency must be set significantly higher than the highest expected frequency within the signal. This elevated sampling frequency guarantees the faithful capture of rapid changes in the current, such as fast transients or appliance switching operations. However, it's important to note that working directly with this type of data demands a high computational cost [64].

2.3.2 V-I Trajectories

V-I trajectories are defined as mutual locus of instantaneous voltage and current waveforms [69]. Lam et al. in [72] were the first to propose an analysis technique involving the two-dimensional diagram generated from a single cycle of voltage (horizontal axis) and current (vertical axis) during the steady state of a load. Their observations revealed significant differences in V-I trajectories among various types

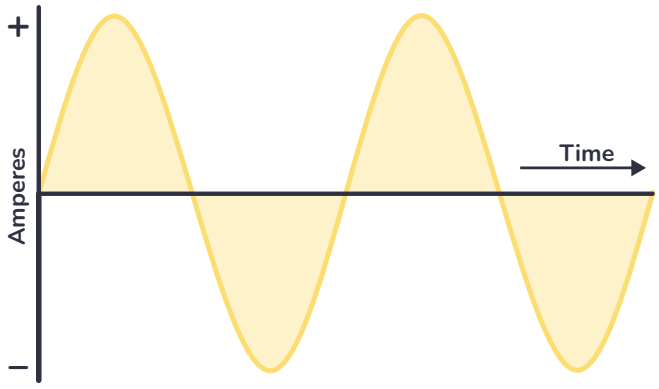


Figure 2.3: Example of raw current waveform.

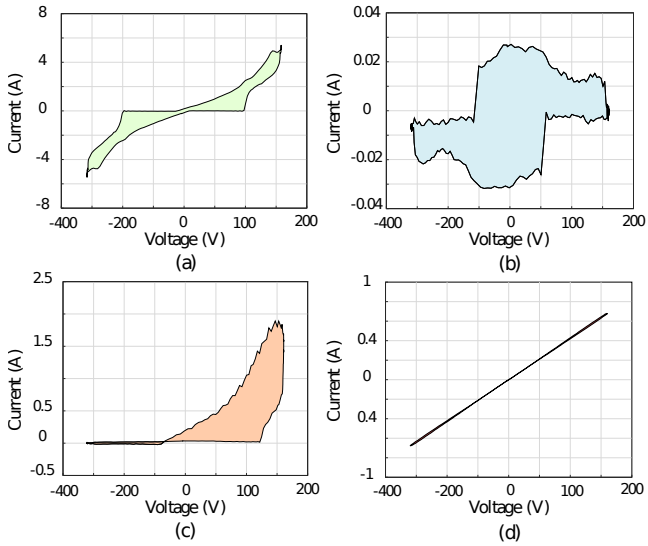


Figure 2.4: V-I trajectories of four loads from the COOL dataset: (a) Drill at full speed (750 W), (b) LED lamp (1.6 W), (c) Drill at half speed (750 W), and (d) Halogen lamp (105 W).

of appliances (loads). For instance, Figure 2.4 illustrates the V-I trajectories of four representative loads from the public COOL dataset, a drill operating at full speed (750 W), a LED lamp (1.6 W), the same drill at half speed (750 W), and a halogen lamp (105 W).

Also, some other features related to the shape of the graph can be extracted from V-I trajectories. This information can be used to differentiate one load from others. Lam and his colleagues proposed some shape features associated with the physical characteristics of a load, such as:

- **Asymmetry:** When a load exhibits varying shapes and peak currents in positive and negative cycles, it manifests as an asymmetrical V-I trajectory.
- **Area:** This characteristic is directly linked to the phase difference between current and voltage.
- **Curvature of Mean Line:** Similar to harmonic distortion, the deviation of the mean line signifies the load's non-linearity.
- **Area of Left and Right Segments:** This feature relates to the time gap between peaks in current and voltage waveforms.
- **Peak of Middle Segment:** Loads with low power consumption typically show a notable peak in the middle segment.

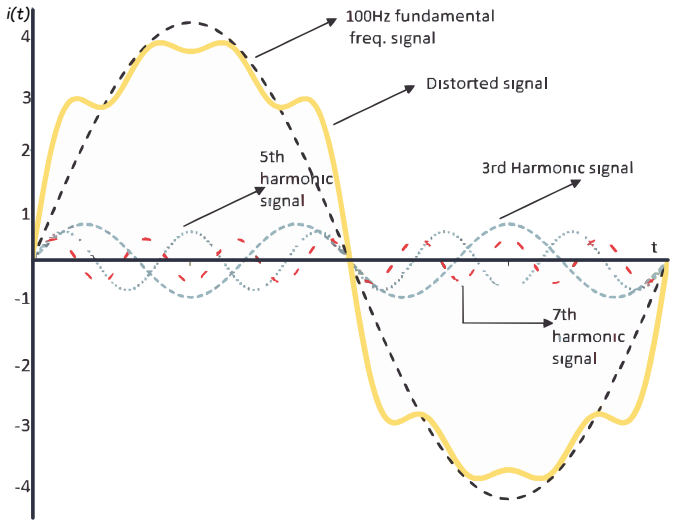


Figure 2.5: Harmonic components of a current waveform.

- **Looping Direction:** This feature correlates with the phase angle between voltage and current. A positive phase angle denotes a clockwise trajectory, while a negative angle indicates the opposite direction.
- **Slope of Middle Segment:** Useful for distinguishing power electronic loads from others, as power electronics typically exhibit a flat middle segment.
- **Self-intersection:** Proportional to the load's harmonic order, this feature accentuates when the load is significantly nonlinear.

2.3.3 Harmonic Components

In electrical power systems, harmonics refer to sinusoidal voltages or currents that have frequencies that are integer multiples of the fundamental frequency (typically 50 or 60 Hz). Harmonics serve as a mathematical description of distortion within voltage or current waveforms, example in Figure 2.5. The term harmonic refers to a component of a waveform that occurs at an integer multiple of the fundamental frequency.

Harmonics are primarily generated by nonlinear loads, such as power electronics, variable frequency drives, rectifiers, and arc furnaces. These devices draw non-sinusoidal currents from the power system, distorting the current signal and introducing harmonics. Current distortion increases losses in cables and other power system components, while voltage distortion impacts the performance of sensitive equipment.

2.3.3.1 Fourier Analysis

The analysis of harmonics is the process of calculating the magnitudes and phases of the fundamental and high order harmonics of the periodic waveforms. The Fourier's theorem states that every non-sinusoidal periodic wave can be decomposed as the sum of sine waves through the application of the Fourier series. This series establishes a relation between a function in the domain of time and a function in the domain of frequency. The equation for the Fourier series can be written as

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty}(a_n cos(n\omega t) + b_n sin(n\omega t)), \tag{2.1}$$

where $f(t)$ is the periodic function representing the waveform, a_0 is the DC component, a_n and b_n are the Fourier coefficients, and ω is the angular frequency. When a signal is decomposed into its constituent sinusoidal components via a Fourier series or transform, the coefficients correspond to the amplitude (magnitude) and phase of each harmonic frequency present in the signal.

2.4 Electrical Parameters at Low-Frequency Sampling Rate

As discussed in Section 2.2.3, in NILM, low-frequency electrical parameters generally refer to data sampled at rates below or comparable to twice the AC base frequency, typically in the range of 1–120 Hz. While these rates are insufficient to fully reconstruct the raw AC waveform, they are adequate for capturing overall trends and energy usage patterns. Importantly, although such parameters are derived from the raw current and voltage signals, the complete waveform itself is not required as input for NILM algorithms. These variables can include RMS voltage and current, active and reactive power, apparent power, power factor, fundamental RMS voltage and current, and others. This subsection will elaborate on these parameters to provide a comprehensive understanding of how they contribute to describing the unique behavior of various loads.

2.4.1 Power

Power refers to the rate at which energy is transferred, converted, or dissipated. It's essentially the amount of work done per unit of time. In electrical terms, power is measured in watts (W) and represents the amount of energy consumed or produced

per second. Actually, 1 watt is equivalent to 1 joule per second (J/s). This concept helps understand how fast energy is used or transmitted in electrical systems. Every day, people come into contact with this electrical parameter, often without realizing it. For example, consider two coffee makers: one rated at 800 watts and another at 1200 watts. The 1200 watts coffee maker produces hotter coffee due to its higher power input compared to the 800 watts model. If one household uses an 800 watts coffee maker while another uses a 1200 watts coffee maker, both brewing the same amount of coffee, the household with the 1200 watts coffee maker will produce hotter coffee due to its higher power consumption. Consequently, it will consume more electricity and may result in a slightly higher energy bill compared to the household using the 800 watts coffee maker.

2.4.2 Instantaneous Power

Instantaneous power is defined as the product of the instantaneous voltage supplied to a device and the instantaneous current flowing through it [73]. It is expressed as

$$p(t) = v(t)i(t). \tag{2.2}$$

In a general scenario where a sinusoidal source provides energy to an arbitrary combination of circuits, the voltage and current are described by Eqs. (2.3) and (2.4):

$$v(t) = V_m \cos(\omega t + \theta_v), \tag{2.3}$$

$$i(t) = I_m \cos(\omega t + \theta_i). \tag{2.4}$$

Here, V_m and I_m represent the peak values of the signals, while θ_v and θ_i denote the phase angles of voltage and current, respectively. Substituting Eqs. (2.3) and (2.4) into Eq. (2.2) yields the following expression for instantaneous power:

$$p(t) = \tfrac{1}{2} V_m I_m \cos(\theta_v - \theta_i) + \tfrac{1}{2} V_m I_m \cos(2\omega t + \theta_v + \theta_i). \tag{2.5}$$

2.4.3 Active Power

Measuring instantaneous power precisely at a given moment is difficult. However, active power, obtained by averaging instantaneous power over a period, is more easily measured. It is defined as

$$P = \frac{1}{T}\int_0^T p(t)\,dt, \tag{2.6}$$

where $p(t)$ denotes the instantaneous power [74]. By substituting Eq. (2.5) into Eq. (2.6), the expression for active power becomes

$$\begin{aligned} P &= \frac{1}{T}\int_0^T \tfrac{1}{2}V_m I_m \cos(\theta_v - \theta_i)\,dt + \frac{1}{T}\int_0^T \tfrac{1}{2}V_m I_m \cos(2\omega t + \theta_v + \theta_i)\,dt, \\ P &= \tfrac{1}{2}V_m I_m \cos(\theta_v - \theta_i)\frac{1}{T}\int_0^T dt + \tfrac{1}{2}V_m I_m \frac{1}{T}\int_0^T \cos(2\omega t + \theta_v + \theta_i)\,dt. \end{aligned} \tag{2.7}$$

The first integral results in a constant, while the second corresponds to a sinusoidal signal. Since the average value of a sinusoid over one period is zero, the second term in Eq. (2.7) vanishes, yielding

$$P = \tfrac{1}{2}V_m I_m \cos(\theta_v - \theta_i). \tag{2.8}$$

In summary, instantaneous power $p(t)$ varies with time, whereas active power P depends on the product of the voltage and current peak values, scaled by the cosine of their phase angle difference [75].

2.4.4 RMS Voltage and Current

To continue with the power definitions, it is necessary to introduce the concept of the effective value, applicable to both voltage and current signals.

The effective value represents the constant magnitude of an alternating current or voltage that produces the same active power dissipation in a resistor as an equivalent direct current or voltage. For instance, consider an alternating current $i(t)$ flowing through a resistor. The instantaneous power dissipated can be obtained from Eq. (2.2), and its average over a full period corresponds to the active power. If instead a direct current (DC) is applied to the same resistor and adjusted until it dissipates the same amount of active power, the resulting DC magnitude equals the effective value of the original AC current.

The effective values of voltage and current are given by

$$V_{rms} = \sqrt{\tfrac{1}{T}\int_0^T v^2(t)\,dt}, \tag{2.9}$$

$$I_{rms} = \sqrt{\frac{1}{T}\int_0^T i^2(t)\,dt}. \tag{2.10}$$

From Eqs. (2.9) and (2.10), it is clear that the effective values correspond to the root-mean-square of the voltage and current signals, which is why they are commonly referred to as RMS voltage and RMS current.

In practice, alternating voltage and current are usually expressed in RMS form, particularly in power analysis. For example, the active power obtained in Eq. (2.8) can also be expressed in terms of RMS quantities as

$$P = V_{rms} I_{rms} \cos(\theta_v - \theta_i). \tag{2.11}$$

2.4.5 Apparent Power

By examining the active power in terms of effective values, as shown in Eq. (2.11), two additional quantities can be derived: the apparent power (S) and the power factor (pF). The apparent power is expressed as

$$S = V_{rms} I_{rms}, \tag{2.12}$$

where S is defined as the product of the RMS voltage and the RMS current. It is referred to as *apparent* because, by analogy with resistive DC circuits, power seems to be obtained directly from the multiplication of voltage and current. To distinguish it from active power, the units of apparent power are expressed in volt-amperes (VA) rather than watts.

Another parameter derived from active power is the power factor (pF), given by

$$pF = \frac{P}{S} = \cos(\theta_v - \theta_i). \tag{2.13}$$

The power factor corresponds to the cosine of the phase difference between voltage and current, which is equivalent to the cosine of the load impedance angle. Its value depends on the nature of the load: for a purely resistive load, voltage and current are in phase, resulting in a power factor of one. Conversely, for a purely reactive load, the phase difference is 90°, and the power factor becomes zero [76].

2.4.6 Reactive and Complex power

To understand reactive power, it is first necessary to introduce the concept of complex power ($\mathbf{S}$), defined as the product of the RMS voltage phasor and the conjugate of the

RMS current phasor. As with any complex quantity, complex power has a real part and an imaginary part:

$$\begin{aligned}\mathbf{S} &= V_{rms} I_{rms} \angle(\theta_v - \theta_i) \\ &= V_{rms} I_{rms} \cos(\theta_v - \theta_i) + j V_{rms} I_{rms} \sin(\theta_v - \theta_i).\end{aligned} \tag{2.14}$$

The real part corresponds to the active power,

$$P = V_{rms} I_{rms} \cos(\theta_v - \theta_i), \tag{2.15}$$

while the imaginary part defines the reactive power,

$$Q = V_{rms} I_{rms} \sin(\theta_v - \theta_i). \tag{2.16}$$

Therefore, complex power can also be expressed as

$$\mathbf{S} = P + jQ. \tag{2.17}$$

Active power P represents the average power delivered to the load and is the only component dissipated usefully. Reactive power Q, on the other hand, quantifies the exchange of energy between the source and the reactive elements of the load. Its unit is the volt-ampere reactive (VAR) [73].

2.4.7 The Power Triangle

To simplify the representation of complex power, the power triangle was developed as a graphical and straightforward method to illustrate the relationship between all power components. It requires only two out of the three parameters to determine the missing value using trigonometric properties, Figure 2.6.

Power analysis reveals that when the reactive power is zero, the loads are purely resistive with a unity power factor. A reactive power less than zero indicates primarily capacitive loads, exhibiting a leading power factor. Conversely, a positive reactive power implies predominantly inductive loads causing a lagging power factor [75].

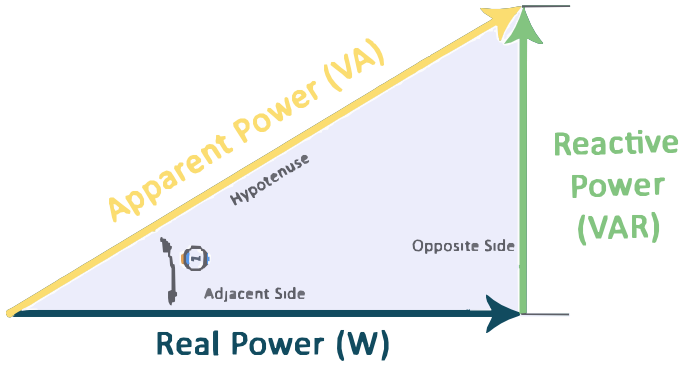


Figure 2.6: Power triangle.

2.5 Artificial Intelligence, Machine Learning and Deep Learning

Artificial Intelligence (AI), Machine Learning (ML), and Deep Learning (DL) are terms often used interchangeably, but they represent a distinct area within the broader field of advanced computing. Understanding the relationships and boundaries between these concepts is essential for grasping the scope and capabilities of each. The Euler diagram in Figure 2.7 illustrates how these fields overlap and interact.

2.5.1 Artificial intelligence

There is no single, universally accepted definition for Artificial Intelligence; however, the Oxford English Dictionary defines AI as "the capacity of computers, or other machines, to exhibit intelligent behavior." This implies that AI systems can seem to think, learn, and act like humans and, in some cases, exceed human capabilities. Such systems are able to analyze vast amounts of data, solve complex problems, make decisions, and perform creative tasks [77].

Given this definition, the following question arises: can a machine truly think? To explore this, Alan Turing proposed the Turing Test in 1950. A computer passes this test if, after a series of written questions, a human interrogator cannot determine whether the responses are from a human or a machine. Achieving this level of performance requires the computer to possess the following capabilities:

- natural language processing to communicate successfully in a human language;
- knowledge representation to store what it knows or hears;
- automated reasoning to answer questions and to draw new conclusions;

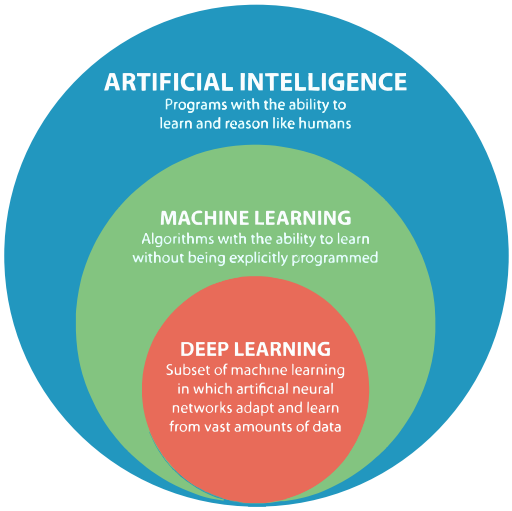


Figure 2.7: Relationship between artificial intelligence, machine learning and deep learning.

- machine learning to adapt to new circumstances and to detect and extrapolate patterns.

Developing these capabilities poses substantial technical and philosophical challenges, pushing the boundaries of computational design and ethical considerations. For instance, natural language processing must handle the subtleties and ambiguities of human language, while knowledge representation demands methods to capture and structure vast and varied information. The Turing Test, though influential, also raises ongoing debate about whether "thinking" can be accurately measured through behavior alone, without understanding the underlying processes [78]

2.5.2 Machine Learning

Machine learning is a branch of AI, originally defined in the 1950s by AI pioneer Arthur Samuel as “the field of study that gives computers the ability to learn without explicitly being programmed”. Machine learning shifts from traditional programming methods by allowing computers to develop their own solutions based on patterns they detect in data. In traditional programming, instructions are carefully set to dictate every step a computer must follow. This approach, however, is often impractical for complex tasks such as recognizing faces in images. Although these tasks are simple for humans, it is very challenging to define them as a set of rules. Machine learning overcomes this by allowing computers to "learn" from data, adapting their responses based on the examples they are given [79].

The ML process begins with data such as text, images, time series from sensors, or sales records prepared as training data. This information allows a ML model to

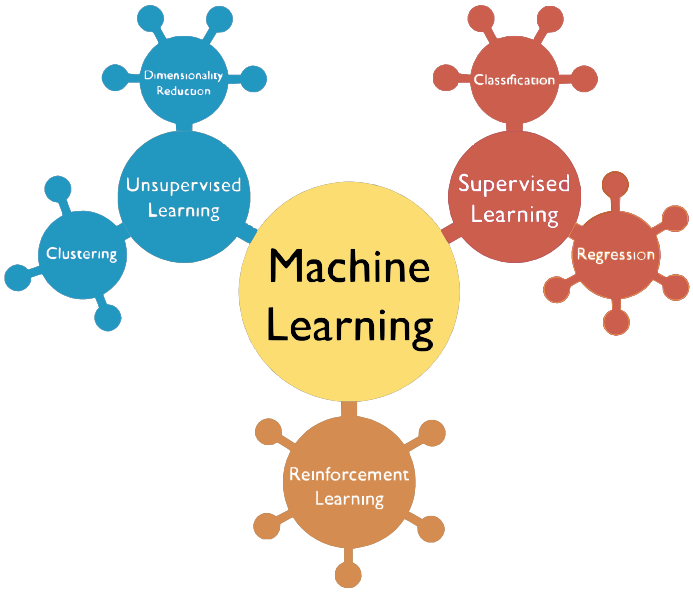


Figure 2.8: Categories of machine learning.

learn by example, with performance improving as more relevant data is provided. The selected model, trained on this data, identifies patterns or makes predictions based on these inputs. By holding back a portion of the data as evaluation data, the model's accuracy can be tested on new, unseen examples, ensuring it generalizes well to real-world scenarios [80].

Machine learning can be further divided into three main types: supervised, unsupervised, and reinforcement learning, Figure 2.8. Each type approaches learning in a unique way, suited to different kinds of tasks.

- Supervised learning: models are trained using labeled datasets, meaning that each piece of data is paired with the correct answer. This helps the model recognize and generalize from examples. For instance, a model could be trained on labeled images of dogs and other objects, allowing it to learn the features that define a dog, so it can later identify similar images on its own. Supervised learning is currently the most widely used approach in the field.

- Unsupervised learning: it works without labeled data. Instead, the model tries to uncover patterns or groupings on its own, revealing insights that may not be immediately obvious. For example, by analyzing sales data, an unsupervised learning model could identify different types of customer behaviors or preferences, even if these groupings weren't pre-defined.

- Reinforcement learning: the model learns through a system of rewards and penalties, encouraging it to make optimal choices through trial and error. This

type of learning is often used in applications like game playing or training autonomous vehicles. By rewarding correct actions, the model gradually learns strategies that maximize the desired outcomes.

2.5.3 Deep Learning

Within the field of machine learning, various models are tailored to suit specific types of tasks. One of the most prominent models is the Artificial Neural Networks (ANNs), which is designed to mimic the way neurons in the human brain operate. ANNs are particularly powerful for complex tasks like image and speech recognition, where traditional algorithms often struggle. For an in-depth look at how ANNs function and their unique advantages, see section 2.6.2.

Based on the concept of ANNs, deep learning is a specialized area of ML that leverages complex, multi-layered neural networks. In deep learning, models are composed of multiple layers between the input and output layers, allowing them to process and extract higher-level features from vast amounts of data. This depth enables deep learning models to excel in tasks that involve intricate patterns, such as natural language processing, image recognition, and autonomous driving [81]. The overall flow of this process is illustrated in Figure 2.9.

By stacking layers, each one captures progressively more abstract representations of the data, deep learning models can achieve impressive accuracy and flexibility in handling complex tasks. The field of deep learning has seen significant advancements due to increased computational power and large datasets, enabling breakthroughs across industries where nuanced data patterns are essential [82].

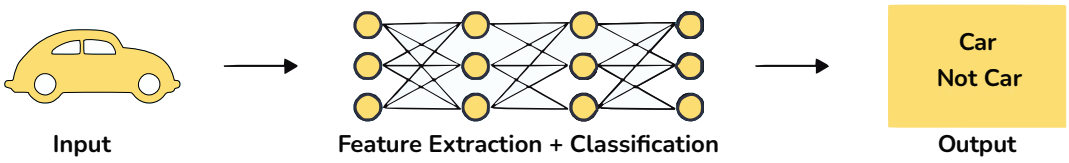


Figure 2.9: Example of deep learning flow.

2.6 Overview of Machine Learning and Neural Network Models

2.6.1 K-Means Clustering

K-means is an unsupervised machine learning algorithm used for clustering. Its goal is to partition a set of data points into k distinct clusters, where each data point belongs to the cluster with the nearest mean, serving as a prototype of the cluster.

The algorithm is particularly useful in exploratory data analysis, image segmentation, and pattern recognition. The image provided in Figure 2.10 illustrates the effect of the K-means algorithm, showing the distribution of data points before and after clustering [83].

This technique works as follows:

1. Initialization: the first step in applying the K-means algorithm is to define the number of clusters, k. This parameter specifies the number of groups into which the data will be divided.

2. Random Centroids: centroids act as the initial representative points for each cluster. Random initialization helps to break symmetry and allows the algorithm to explore various configurations before converging on a solution.

3. Distances Calculation: for each data point x_i, the distance to each centroid μ_j $(where j = 1, 2, \ldots, k)$ has to be calculated. In K-means, the most commonly used distance metric is the Euclidean distance, defined as follows:

$$d(x_i, \mu_j) = \sqrt{\sum_{m=1}^{M} (x_{i,m} - \mu_{j,m})^2}, \tag{2.18}$$

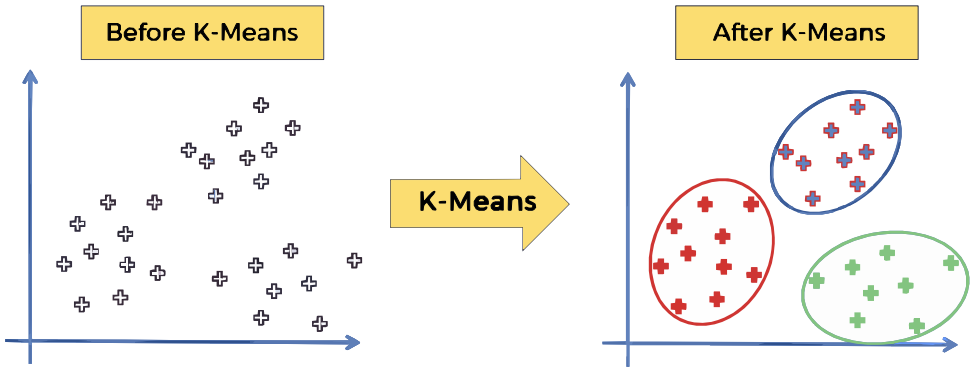


Figure 2.10: Demonstration of K-means application.

where M represents the number of dimensions of each data point.

4. Assignation of data points to clusters: Each data point x_i is assigned to the cluster whose centroid is closest, thereby forming k clusters. Formally, this is achieved by selecting the cluster j that minimizes the distance:

$$Cluster(x_i) = \arg\min_j d(x_i, \mu_j) \tag{2.19}$$

5. Centroids update: after the initial assignment of points to clusters, the centroids are recalculated. For each cluster C_j, the new centroid μ_j is computed as the mean of all data points within that cluster:

$$\mu_j = \frac{1}{|C_j|} \sum_{x \in C_j} x, \tag{2.20}$$

where $| C_j |$ denotes the number of data points in cluster j, and $x \in C_j$ indicates that x belongs to cluster j.

6. Loop: Steps 3 to 5 are repeated iteratively until convergence. Convergence is typically reached when the centroids stabilize, meaning there is no significant change in their positions, or when the assignment of data points to clusters remains unchanged.

The K-means algorithm optimizes the clustering by minimizing the within-cluster sum of squares, also known as inertia J. The objective function is given by:

$$J = \sum_{j=1}^{k} \sum_{x_i \in C_j} \|x_i - \mu_j\|^2. \tag{2.21}$$

2.6.2 Artificial Neural Networks

Artificial Neural Networks (ANNs) are computational models inspired by the structure and functioning of the human brain. They consist of interconnected nodes, or neurons, organized in layers that process and transform input data to produce a desired output. ANNs are foundational to modern machine learning and have demonstrated exceptional capabilities in tasks such as image recognition, natural language processing, and complex data analysis. By adjusting the connections between neurons, ANNs learn patterns within the data, allowing them to generalize beyond the training samples and make accurate predictions on new data [82].

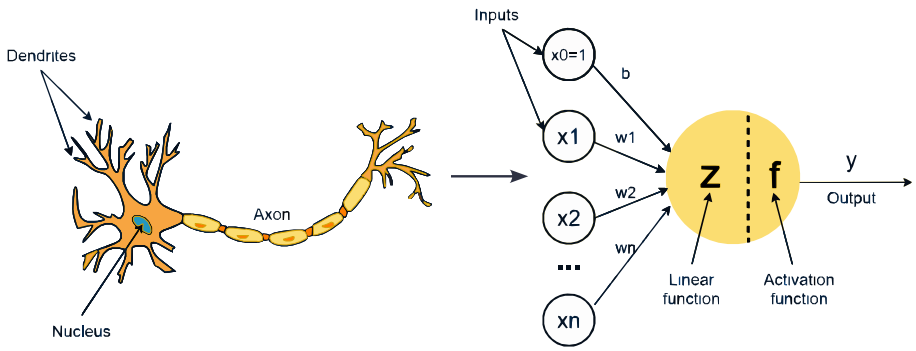


Figure 2.11: Perceptron structure.

2.6.2.1 The Perceptron: Input-Output Structure

The perceptron is the simplest form of a neural network model and serves as a building block for more complex networks. A perceptron, which can be seen in Figure 2.11, consists of a single neuron that receives multiple input values, applies a weight to each input, sums the weighted inputs, and passes this sum through an activation function to produce an output [84]. For a perceptron with n inputs $x_1, x_2, x_3, ..., x_n$ and corresponding weights $w_1, w_2, w_3, ..., w_n$ the output the output y is computed as:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right), \tag{2.22}$$

where w_i are the weights associated with each input x_i, b a bias term, and f is the activation function. The role of the bias b is to shift the activation function, allowing the perceptron to model patterns that do not pass through the origin. When multiple perceptrons are connected in layers, they form the basis of an artificial neural network.

2.6.2.2 ANN Structure and Layers

An ANN typically comprises multiple layers of neurons: an input layer, one or more hidden layers, and an output layer, Figure 2.12. This layered structure enables the network to capture complex, hierarchical patterns in the data.

- Input Layer: the input layer is responsible for receiving raw data. Each neuron in this layer corresponds to a feature in the input data, and no computation occurs at this stage.
- Hidden Layers: hidden layers sit between the input and output layers and perform the actual processing of information. Each neuron in a hidden layer receives inputs from the neurons in the preceding layer, applies weights, and produces an output

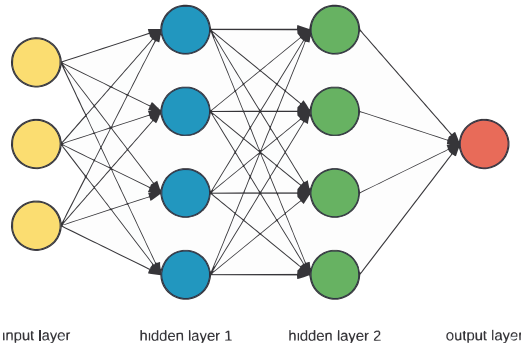


Figure 2.12: Artificial Neural Network basic architecture.

through an activation function. The number and size of hidden layers determine the capacity of the network to learn complex patterns.

- Output Layer: the output layer produces the final predictions of the network. The structure of the output layer depends on the type of task: for classification, the output layer often has one neuron per class with a softmax activation, while for regression tasks, a single linear output neuron may suffice.

If the output layer has n neurons, the equation representing the output of an ANN with L layers must account for each output neuron separately. The output vector of the network, represented as $y = [y_1, y_2, \ldots, y_n]$ where each y_j is the output of the j_{th} neuron in the output layer, can be expressed as follows:

$$y = f_L\left(W_L \cdot f_{L-1}\left(W_{L-1} \cdots f_1\left(W_1 \cdot \mathbf{x} + \mathbf{b}_1\right) + \mathbf{b}_{L-1}\right) + \mathbf{b}_L\right), \tag{2.23}$$

W_i and $\mathbf{b}_i$ are the weight matrix and bias vector for layer i, f_i is the activation function applied element-wise in layer i, $\mathbf{x}$ is the input vector, and $\mathbf{y}$ the output vector with n components, each corresponding to an output neuron.

2.6.2.3 Activation Functions

The activation function f introduces non-linearity into the network, enabling it to model complex relationships within the data. Without non-linear activation functions, a neural network would behave as a linear model, regardless of the number of layers. Common activation functions include the Sigmoid, Hyperbolic Tangent (tanh), Rectified Linear Unit (ReLU), and Softmax [85].

- **Sigmoid:** This function compresses input values into the range $(0, 1)$, making it particularly suitable for use in the output layer of binary classification problems.

The sigmoid activation function is defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{2.24}$$

- **Hyperbolic Tangent (tanh):** The tanh function maps input values to the range $(-1, 1)$, providing a zero-centered output that is often beneficial in hidden layers. It is expressed as follows:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}. \tag{2.25}$$

- **Rectified Linear Unit (ReLU):** As shown in the equation below, this function outputs the input directly if it is positive; otherwise, it returns zero. ReLU is computationally efficient and alleviates the vanishing gradient problem.

$$ReLU(x) = \max(0, x) \tag{2.26}$$

- **Softmax:** This function is commonly used in the output layer for multi-class classification tasks. It converts a vector of raw scores (logits) into a probability distribution over predicted output classes, with the probabilities summing to 1. Each value represents the likelihood of the input belonging to a particular class. The Softmax function for a vector z of length K is defined as:

$$Softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \quad i = 1, 2, ..., K. \tag{2.27}$$

2.6.2.4 Backpropagation

Backpropagation is a learning algorithm used to adjust the weights and biases in an ANN to minimize the error between the predicted and actual outputs. It consists of two phases:

- Forward Pass: The network performs a forward pass to compute the predicted output based on the current weights and biases.
- Backward Pass: Using the error between the predicted output and the true output, the network computes the gradient of the loss function with respect to each weight. This phase applies the chain rule of calculus to propagate the error backwards through the network.

The objective of backpropagation is to minimize a loss function L, commonly the Mean Squared Error (MSE) for regression or Cross-Entropy for classification. To minimize

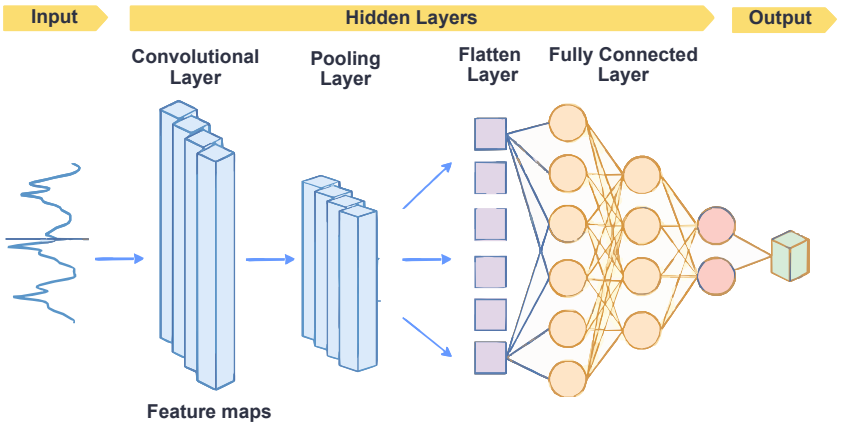


Figure 2.13: Convolutional Neural Network basic architecture.

the loss function, the gradient descent algorithm is typically employed. This iterative optimization method updates the weights w of the model based on the computed gradients of the loss function with respect to those weights. The update rule is given by:

$$w_{ij} = w_{ij} - \eta \frac{\partial L}{\partial w_{ij}}, \tag{2.28}$$

where w_{ij} is the weight connecting neuron j in one layer to neuron i in the following layer, η is the learning rate, controlling the size of the weight updates, and $\frac{\partial L}{\partial w_{ij}}$ is the partial derivative of the loss function with respect to w_{ij}.

2.6.3 1D Convolutional Neural Network

A 1D Convolutional Neural Network (1D CNN) is a specialized type of ANN that is particularly effective for sequence data analysis. Each neuron processes a small segment of the input data, applying filters over overlapping "windows" across the input sequence. This structure allows the network to capture localized patterns, making it well-suited for structured data such as time series, text sequences, and sensor readings. Like other ANNs, 1D CNNs are trained using the backpropagation method, which iteratively adjusts weights and biases to optimize the model's feature detection within sequential data [86].

2.6.3.1 Structure of a 1D CNN

A typical 1D CNN consists of a sequence of convolutional layers, followed by pooling layers, and ends with fully connected layers (see Figure 2.13). Each layer has a specific role in processing data and learning features.

- Convolutional Layer: This layer applies a set of filters (or kernels) to the input data to detect features in small segments. Each filter slides along the input data, performing a convolution operation.
- Pooling Layer: Pooling layers downsample the data by summarizing information within each segment. This reduces dimensionality and helps control overfitting.
- Flatten Layer: Transforms the multi-dimensional feature maps into a single one-dimensional vector, making the data suitable for the fully connected layers.
- Fully Connected Layer: Once the convolutional and pooling layers have extracted features, fully connected layers process this feature map for classification or regression tasks.

2.6.3.2 Convolution Operation

In a 1D CNN, the convolution operation detects patterns along a single dimension, typically spatial or temporal, depending on the data type. For an input sequence $x = [x_1, x_2, \ldots, x_n]$ with n elements, a filter $w = [w_1, w_2, \ldots, w_k]$ of length k (also called the kernel size) slides over x, producing a feature map h, where each element h_i in the feature map is computed as:

$$h_i = \sum_{j=1}^{k} w_j \cdot x_{i+j-1} + b, \tag{2.29}$$

w_j are the weights of the filter x_{i+j-1} represents the segment of the input over which the filter is applied, b is a bias term added to the output.

The filter slides over the input sequence with a stride s, controlling how many steps the filter moves at each position. The stride affects the output size, as a larger stride results in fewer elements in the output feature map.

2.6.3.3 Padding

Padding is often used to control the size of the output feature map. There are two main types:

- Valid Padding: No padding is applied, resulting in an output feature map that is smaller than the input.
- Same Padding: Padding is added to the input sequence (usually with zeros) to ensure the output feature map has the same length as the input.

With padding p, the size of the output feature map h for an input of length n, filter size k, and stride s is given by:

$$Outputsize = \frac{s + 2p - k}{n} + 1. \tag{2.30}$$

2.6.3.4 Pooling Layers

Pooling layers reduce the dimensionality of the feature map by summarizing information in each segment. For 1D CNNs, common pooling methods include:

- Max Pooling: Takes the maximum value in each segment, retaining the strongest feature.
- Average Pooling: Takes the average value of each segment, providing a smoothed representation.

For a pooling layer with a pooling size p and stride s, the pooling operation on feature map h produces a downsampled output $h\prime$ where each element h'_i for max pooling is calculated as:

$$h'_i = \max(h_{i \cdot s}, h_{i \cdot s+1}, \ldots, h_{i \cdot s+p-1}) \tag{2.31}$$

2.6.3.5 Fully Connected Layers

After the convolutional and pooling layers, the feature map has to be flattened into a single vector and passed through one or more fully connected layers. Each neuron in a fully connected layer has connections to every neuron in the previous layer. This structure enables the network to learn complex mappings from the features extracted by convolutional layers to the final output.

The output y of a fully connected layer with input $\mathbf{x}$, weights W, and bias b is:

$$y = f(W \cdot \mathbf{x} + b). \tag{2.32}$$

2.6.4 Recurrent Neural Networks

A Recurrent Neural Network (RNN) is a neural network architecture for sequential data processing, such as time series or language. Unlike traditional feedforward neural networks, RNNs have connections that loop back, enabling them to retain information across time steps. This loop allows the network to maintain a "memory" of previous inputs, which is essential for capturing temporal dependencies within a sequence. The RNN architecture consists of three primary components: an input layer, which receives

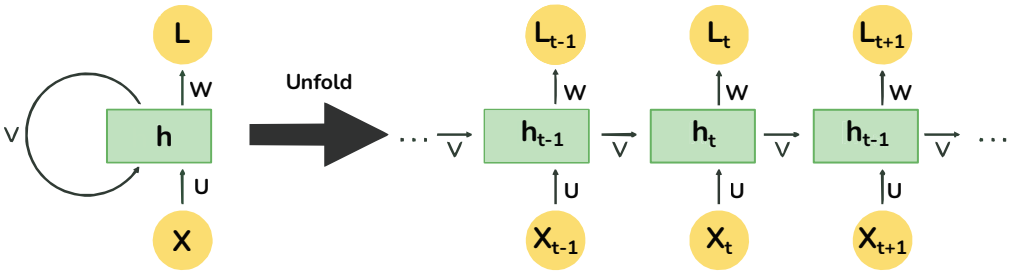


Figure 2.14: Recurrent Neural Networks basic architecture.

the sequential data; a hidden layer with recurrent connections that processes and retains information from previous steps; and an output layer, which produces the final predictions [87]. Figure 2.14 provides an overview of this standard structure.

2.6.4.1 RNN input layer

At each time step t, an RNN receives an input vector x_t from a sequence $\{x_1, x_2, \ldots, x_T\}$, where T represents the total length of the sequence. The input vector x_t could represent various forms of data depending on the application. For example, in a natural language processing task, x_t might represent a word embedding for a word in a sentence, while in a time series application, x_t could be a scalar or vector containing features for a specific timestamp.

2.6.4.2 Hidden State and Recurrent Connections

The hidden state is a central component of RNNs, providing memory that enables the network to capture dependencies across time steps. At each time step t, the hidden state h_t is updated based on the current input x_t and the hidden state from the previous time step h_{t-1}. This recurrence relation is typically represented as:

$$h_t = f(W_{hx} \cdot x_t + W_{hh} \cdot h_{t-1} + b_h), \tag{2.33}$$

where the hidden state vector h_t at time t represents the memory of the network. The hidden state is updated by combining the input vector x_t, connected to the hidden layer via the weight matrix W_{hx}, and the previous hidden state h_{t-1}, connected through the recurrent weight matrix W_{hh}. Additionally, a bias vector b_h is applied to the hidden state. The non-linear activation function f, commonly chosen as $tanh$ or $ReLU$, is then applied to introduce non-linearity and stabilize gradients during training.

The recurrent nature of W_{hh} is what enables the RNN to maintain information across the sequence, as each hidden state h_t is influenced by both the current input and

the accumulated knowledge from previous steps.

2.6.4.3 Output Layers

The output of an RNN at each time step can vary based on the task. In a standard RNN, the output y_t at each time step t is computed as:

$$y_t = g(W_{hy} \cdot h_t + b_y), \tag{2.34}$$

the output vector at time t, denoted by y_t, represents the network's prediction. This output is generated by connecting the hidden state h_{th} to the output layer through the weight matrix W_{hy} and applying an output bias b_y. The activation function for the output, represented by g, is chosen based on the task requirements; it may be a softmax function for classification tasks or a linear activation for regression tasks.

2.6.4.4 Unfolding Mechanism

The recurrent nature of RNNs can be visualized through an unfolding process, Figure 2.14, where the network is "unfolded" over time to reveal a chain structure of connected layers, each corresponding to one time step in the sequence. When unfolded, the RNN resembles a deep neural network where each layer represents a single time step t.

To compute the output sequence, the model iteratively applies the recurrence relation:

1. Initialize the hidden state h_0, typically to a vector of zeros or small random values.
2. For each time step $t = 1, \ldots, T$:
   (a) Compute the hidden state h_t using Eq. 2.33
   (b) Compute the output y_t with Eq. 2.34.

Thus, the unfolding allows the RNN to propagate information forward through the sequence while adjusting the hidden state at each time step based on the input and the previous hidden state.

2.6.4.5 Backpropagation Through Time (BPTT)

Training RNNs requires a specialized form of backpropagation known as Backpropagation Through Time (BPTT), which accounts for sequential dependencies by backpropagating errors across time steps. During BPTT, the loss for each time

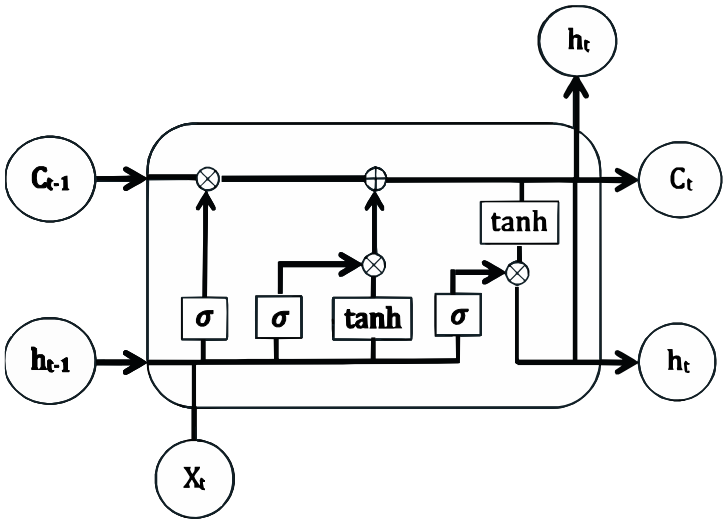


Figure 2.15: Long Short-Term Memory cell.

step t is computed, and gradients are calculated with respect to each weight parameter across all steps. The gradient is then used to adjust the weights W_{hx}, W_{hh}, and W_{hy}.

However, due to the recurrent nature of RNNs, BPTT can encounter vanishing or exploding gradient problems, especially over long sequences. The vanishing gradient issue occurs because repeated multiplication by small gradients causes gradients to shrink exponentially, making it difficult for the model to learn long-term dependencies. In contrast, the exploding gradient issue arises when gradients grow excessively, which can destabilize training.

2.6.5 Long Short Term Memory networks

Long Short-Term Memory networks are an advanced form of RNNs that overcome the limitations of standard RNNs in capturing long-term dependencies. The core component of an LSTM is the *memory cell*, which is capable of retaining information over extended sequences, enabling the network to handle long dependencies effectively [88]. The LSTM cell is shown in Figure 2.15, where each cell contains a series of gates such as the input gate, forget gate, and output gate. These gates regulate the flow of information, making it possible for the network to selectively retain or discard information at each time step t.

2.6.5.1 Memory Cell and Cell State

At the heart of the LSTM is the cell state, represented by ct in the diagram. The cell state flows horizontally across time steps and serves as the main avenue for information

retention. Unlike the hidden state in a basic RNN, the cell state in an LSTM is designed to preserve long-term information with minimal modifications, thanks to the gating mechanisms that control its updates.

2.6.5.2 Forget Gate

The first gate, represented by the symbol (sigmoid activation function), is the forget gate, which decides how much of the previous cell state c_{t-1} should be retained. It takes as inputs the previous hidden state h_{t-1} and the current input x_t, applies a set of weights W_f, and outputs a value between 0 and 1 for each element in c_{t-1}. If the forget gate outputs a value close to 1, that part of the previous cell state is retained, whereas a value close to 0 causes the network to forget it. This gating mechanism is crucial for eliminating irrelevant information from earlier time steps and is computed as follows:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f). \tag{2.35}$$

2.6.5.3 Input Gate and Candidate Cell State

The input gate decides how much of the new information should be incorporated into the cell state. The gate uses a sigmoid activation function to generate values between 0 and 1, acting as a filter that controls the flow of information. At the same time, a candidate cell state $\tilde{c}_t$ is computed through a hyperbolic tangent ($tanh$) activation, representing the potential new content that could be stored in the memory cell. The interaction between these two components is described by the following equations:

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right), \tag{2.36}$$

$$\tilde{c}_t = \tanh\left(W_c \cdot [h_{t-1}, x_t] + b_c\right), \tag{2.37}$$

where i_t is the input gate activation, $\tilde{c}_t$ is the candidate cell state. The updated cell state ct is then computed by combining the previous cell state (scaled by the forget gate) and the candidate cell state (scaled by the input gate):

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t. \tag{2.38}$$

2.6.5.4 Output Gate and Hidden State

The output gate controls what portion of the cell state ct is sent to the hidden state ht, which serves as both the output of the current cell and the input for the next time

step. Like the other gates, the output gate takes h_{t-1} and x_t as inputs and applies a sigmoid activation to determine which parts of c_t will contribute to the next hidden state. The hidden state is then calculated by applying a tanh activation to the cell state and scaling it by the output gate:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \tag{2.39}$$

$$h_t = o_t \cdot tanh(c_t). \tag{2.40}$$

Each of the above stages ensures that the LSTM cell maintains relevant information over long sequences, while also dynamically adapting to incoming data at each time step. The interactions of these gates enable the LSTM to decide what to remember, what to forget, and what to output based on the context, which makes it especially useful for applications like language modeling and sequence prediction. The diagram you provided visually represents these components and how they interact in each LSTM cell, showcasing the flow of information through the gates and the update of both the cell and hidden states.

2.6.6 Evaluation Metrics

Evaluating the performance of a neural network, particularly in classification tasks, requires more than just accuracy. Several metrics offer insights into the model's behavior in terms of false positives, false negatives, and overall balance between precision and recall.

- **Accuracy:** Accuracy is the ratio of correctly predicted observations to the total number of observations. It provides a general idea of how well the model is performing.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.41}$$

- **Precision:** Precision measures the proportion of positive identifications that were actually correct.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.42}$$

- **Recall:** Also known as sensitivity or true positive rate, recall measures the proportion of actual positives that were identified correctly.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.43}$$

- **F1-score:** The F1-score is the harmonic mean of precision and recall, offering a balance between the two.

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.44}$$

- **ROC Curve:** The Receiver Operating Characteristic (ROC) curve is a graphical plot used to assess the diagnostic ability of a binary classifier system as its discrimination threshold is varied. It plots the True Positive Rate (Recall) against the False Positive Rate (FPR), defined as:

$$\text{FPR} = \frac{FP}{FP + TN} \tag{2.45}$$

  A model with perfect classification has an ROC curve that passes through the top-left corner, indicating high true positive and low false positive rates.

These metrics are essential in domains where class imbalance exists or where the cost of false positives and false negatives differs significantly. Their correct interpretation enables the selection and tuning of models more effectively [89].

2.7 Internet of Things (IoT)

The Internet has become a fundamental element in the current era of global digital integration. In this technological context, a wide range of human activities can now be interconnected through the network, giving rise to what is known as the Internet of Things or IoT.

The IoT can be broadly defined as the interaction between the physical and digital worlds, where the latter connects to the former through sensors and actuators [90]. A more formal definition describes IoT as a dynamic global network infrastructure with self-configuring capabilities, based on interoperable communication protocols and standards. In this context, both physical and virtual "things" possess unique identities, physical characteristics, and virtual personalities. These entities operate using intelligent interfaces and are seamlessly integrated into the information network [91].

Based on this conceptualization, several key characteristics are generally shared among systems developed under the IoT paradigm [92]:

- **Dynamism and Self-adaptation:** IoT devices and systems are capable of dynamically adapting to changing conditions in their operational environment.

They can respond autonomously based on contextual factors such as environmental data or user behavior. For instance, an automated irrigation system may shut off on its own when rainfall is detected, demonstrating contextual adaptation based on weather conditions.

- **Self-configuration:** IoT systems are designed with the capacity for autonomous configuration. This feature enables large-scale device coordination without requiring constant manual intervention. Devices are able to structure themselves, establish network connections, and update software independently.
- **Interoperable Communication Protocols:** Multiple communication protocols are implemented to ensure interoperability among heterogeneous devices and with surrounding infrastructure. This supports seamless data exchange across platforms.
- **Unique Identity:** Each component in the system—sensors, actuators, controllers, etc.—must have a unique identifier or address. This enables individual communication, monitoring, configuration, and control at the device level.
- **Integration into the Information Network:** IoT devices are typically embedded into broader information networks, allowing them to communicate and exchange data with other devices and systems. They can be discovered dynamically and provide descriptive information to other devices or user-facing applications. A representative example is a smart energy monitoring network, where smart meters exchange data autonomously.

2.7.1 IoT Architecture

When designing the architecture of an IoT-based system, it is necessary to define all the components involved in the project. This design may vary substantially depending on the specific application and requirements, and thus, no single architecture can be universally applied. Nevertheless, a basic layered architecture is widely accepted within the scientific community, comprising the perception layer, network layer, and application layer.

For more complex implementations, a three-layer architecture may prove insufficient. In response to this, a more elaborate five-layer architecture has been proposed, incorporating additional processing and business layers to complement the original model [93]. The resulting five-layer IoT architecture is depicted in Figure 2.16.

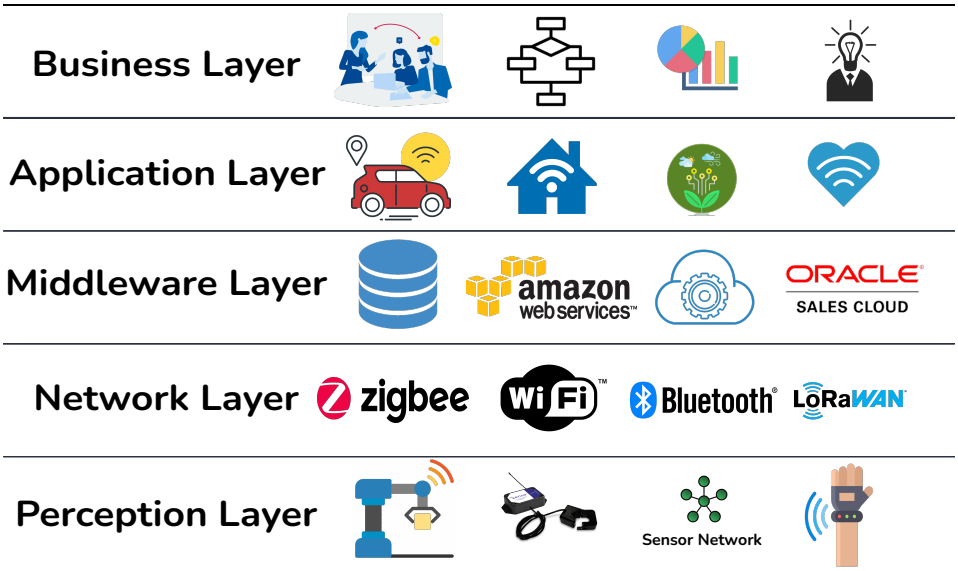


Figure 2.16: Five-layer architecture of the Internet of Things

2.7.1.1 Perception Layer

The perception layer represents the foundational tier of the Internet of Things architecture—essentially, it encompasses the "things" in the IoT paradigm. These "things" refer to all devices that directly interact with their surrounding environment. They are characterized by unique functionalities and are capable of performing various tasks such as sensing, control, and monitoring. These devices may operate in isolation or as part of a wireless sensor network, where they can exchange information with other nodes or collect data from different processes, depending on the specific infrastructure in place. Many of these devices possess the capability to process the captured data locally and present it directly; however, in other cases, the collected data must be transmitted wirelessly to another processing unit or directly to the cloud for further handling [94, 95].

2.7.1.2 Network Layer

As described in the perception layer, the devices responsible for data generation typically communicate wirelessly. To do so, they must implement various communication techniques. The primary function of the connectivity layer is to establish reliable connections among smart instruments, networking devices, and servers, with the ultimate goal of transmitting the data acquired by the perception layer. This layer is fundamentally defined by networking protocols, which, within an IoT system, span across the physical or data link layer, network layer, transport layer,

and application layer. These layers will be described in more detail in the following sections [93, 92, 96].

2.7.1.3 Processing Layer

Once the devices in the perception layer are operational and equipped with some form of communication, they begin to generate large volumes of data. This information must be processed, analyzed, and stored—tasks that are handled by the processing layer, also referred to as the middleware layer. This layer leverages various technologies, including databases, cloud computing, and big data analytics [94].

2.7.1.4 Application Layer

The application layer encompasses all the services that an IoT system can provide. These may include email notifications, actuator control, security management, data monitoring, and more. The utility of the IoT system depends on the domain in which it is deployed, and thus it may be applied across diverse fields such as home automation, smart cities, environmental monitoring, smart grids, logistics, agriculture, and industrial applications [97].

2.7.1.5 Business Layer

The business layer is responsible for the overall management of the IoT system. This includes handling applications, business models, services, and user privacy. Within this layer, decisions are made regarding what information will be generated, how it will be processed, and how it will be presented. Essentially, this layer functions as the global decision-making component of the project [93].

2.7.2 Technologies in the Perception Layer

2.7.2.1 Wireless Sensor Networks (WSN)

Some sensors are capable of connecting independently to the network, meaning they do not rely on other nodes to operate. These devices function as standalone units and transmit their measurements directly to the Internet. However, there are also sensor devices that operate collaboratively, forming what is known as a Wireless Sensor Network (WSN).

A WSN consists of distributed nodes located across various areas to collect information and transmit it collectively through the network to a central node, referred

to as the master or coordinator. This central node acts as the network's gateway, managing the connection to the Internet. Modern WSNs are typically bidirectional, allowing not only data transmission but also remote control of the sensors themselves. These networks are widely used in both industrial and consumer applications, including process monitoring, machine condition supervision, and more. WSNs can be built using various wireless communication technologies such as Bluetooth, Zigbee, or LoRa, with the choice of technology depending primarily on transmission speed and range [98, 99]. Common applications include climate monitoring, air quality sensing, soil moisture tracking, smart grids, and structural health monitoring.

2.7.3 Technologies in the Network Layer

2.7.3.1 Communication Models

- **Request/Response:** A communication model in which a client sends requests to a server, and the server replies. Upon receiving a request, the server determines the appropriate response, retrieves the required data, formats the resource representation, prepares the response, and then sends it back to the client. This is a stateless communication model, where each request-response pair operates independently.

- **Publish/Subscribe:** A communication model involving publishers (data sources), brokers (intermediaries), and subscribers (clients). Publishers send data to specific topics managed by the broker without knowledge of the clients. Clients subscribe to topics through the broker, which then delivers any published data to all subscribed clients.

- **Push-Pull:** In this model, data producers push information to message queues, and consumers pull data from these queues. The producers and consumers operate independently, with the queue acting as a decoupling mechanism and a buffer to handle differences in data production and consumption rates.

- **Exclusive Pair:** A fully bidirectional communication model involving a persistent connection between the client and the server. The connection remains open until explicitly closed by the client. Once the initial setup is complete, both client and server can send messages to one another. This is a stateful communication model in which the server maintains awareness of each active connection.

2.7.3.2 Communication Protocols

As discussed throughout this document, communication is a fundamental component of Internet of Things systems. Given the versatility of the IoT paradigm, multiple communication methods may be implemented depending on the system's intended application. Communication in the network layer is structured according to the link layer, the network/Internet layer, the transport layer, and finally, the application layer. The following sections present the communication techniques commonly used to support tasks specific to each of these layers.

2.7.3.3 Link Layer

In computer networks, the link layer is the lowest layer in the Internet protocol suite. Link layer protocols determine how information is physically transmitted to the network layer on a connected host. The following are some common link layer protocols:

- **802.3 Ethernet:** IEEE 802.3 is a set of standards established by the Institute of Electrical and Electronics Engineers (IEEE) that define Ethernet-based networks, as well as the working group assigned to develop these standards. IEEE 802.3 is also known as the Ethernet standard and defines the physical layer and the medium access control (MAC) sublayer of the data link layer for wired Ethernet networks, typically used in Local Area Networks (LANs). It specifies the physical and network characteristics of Ethernet applications, including how physical connections between nodes (routers/switches/hubs) are made through various wired media such as copper coaxial cable or fiber optics. The technology is designed to work with the IEEE 802.1 standard for network architecture, and its first release was Ethernet II in 1982, featuring 10 Mbit/s over thick coaxial cable [100].

- **802.11 WiFi:** IEEE 802.11 is a set of standards developed by the IEEE that define specifications for implementing Wireless Local Area Networks (WLANs) over various frequency bands, including 900 MHz and the 2.4, 3.6, 5, and 60 GHz bands. The base version of the standard was released in 1997 and has undergone numerous amendments. Although each amendment is technically superseded when incorporated into the latest version of the standard, industry practices often promote these revisions as distinct versions for marketing purposes, which leads to each becoming a de facto standard [100].

- **802.16 WiMAX:** IEEE 802.16 is a family of standards for broadband wireless networks. WiMAX (Worldwide Interoperability for Microwave Access) provides data rates ranging from 1.5 Mb/s to 1 Gb/s. The most recent update (802.16m) offers data rates of up to 100 Mb/s for mobile stations and 1 Gb/s for fixed stations. The specifications are available on the official website of the IEEE 802.16 working group [96].

- **802.15.4 LR-WPAN:** IEEE 802.15.4 defines standards for Low-Rate Wireless Personal Area Networks (LR-WPANs). These standards form the foundation for higher-level communication protocols such as ZigBee. LR-WPANs support data rates ranging from 40 Kb/s to 250 Kb/s and are designed for low-cost, low-speed communication between power-constrained devices. They operate at 868/915 MHz and 2.4 GHz, offering lower and higher data rates respectively [92].

- **2G/3G/4G/5G Mobile Communication:** Several generations of mobile communication standards have been developed, including second-generation (2G, e.g., GSM and CDMA), third-generation (3G, e.g., UMTS and CDMA2000), fourth-generation (4G, e.g., LTE), and fifth-generation (5G), which offers significantly higher data rates (up to 10 Gb/s), ultra-low latency, and massive connectivity, making it particularly suitable for IoT applications that demand real-time data transmission and scalability. Technical details and specifications for these standards are available from the official 3GPP documentation [101].

- **802.15.1 Bluetooth:** Bluetooth is based on the IEEE 802.15.1 standard and is a low-cost, low-power wireless communication technology suitable for short-range (8–10 meters) data exchange between mobile devices. It defines a Personal Area Network (PAN) and operates in the 2.4 GHz band. Depending on the version, data rates range from 1 Mb/s to 24 Mb/s. A low-cost, ultra-low-power variant known as Bluetooth Low Energy (BLE or Bluetooth Smart) was merged into the Bluetooth v4.0 standard in 2010 [96].

- **LoRaWAN R1.0 LoRa:** LoRaWAN is a long-range communication protocol developed by the LoRa™ Alliance, a non-profit open association. It defines a Low Power Wide Area Network (LPWAN) standard for enabling IoT. Its goal is to ensure interoperability among various operators under a global open standard. LoRaWAN supports data rates ranging from 0.3 kb/s to 50 kb/s and operates in ISM bands at 868 and 900 MHz. According to Postscapes, LoRa can communicate over distances up to 48 kilometers in unobstructed environments [94].

2.7.3.4 Network Layer

The network layer has three primary functions: addressing, routing, and path determination. Its goal is to transmit datagrams from a source network to a destination network by interconnecting multiple networks and selecting the most efficient route. Each network has a unique identifier, known as an Internet Protocol (IP) address. The most common IP addressing schemes are IPv4 and IPv6.

- **IPv4:** Internet Protocol version 4 is the most widely used protocol for identifying devices on a network. IPv4 uses a 32-bit addressing scheme, allowing for 2^{32} or 4,294,967,296 unique addresses. Due to the rapid growth in internet-connected devices, these addresses were exhausted by 2011, prompting the transition to IPv6 [100].
- **IPv6:** Internet Protocol version 6 is the successor to IPv4, developed to provide a vastly expanded address space to accommodate the growing number of devices. IPv6 uses a 128-bit address space, offering 2^{128} or approximately 3.4×10^{38} unique addresses [102].
- **6LoWPAN:** Short for IPv6 over Low-Power Wireless Personal Area Networks, 6LoWPAN was developed by the Internet Engineering Task Force (IETF) in 2007. It enables IPv6 communication over IEEE 802.15.4-based low-power wireless sensor networks, making each node individually addressable from the internet [103].

2.7.3.5 Transport Layer

The transport layer enables host-to-host communication regardless of the underlying network. It is primarily responsible for error correction, providing quality and reliability to the end user. The two most common transport layer protocols are TCP and UDP.

- **TCP:** The Transmission Control Protocol is widely used in applications such as web browsing, email, and file transfers. It is a connection-oriented protocol that ensures data packets are delivered in order and without errors. It provides a stable communication service with extensive error handling and flow control to prevent congestion by matching the sender's speed to the receiver's capacity. Due to its overhead and complexity, TCP is generally not suitable for low-power environments, where UDP is often preferred [94].

- **UDP:** The User Datagram Protocol is a connectionless, transaction-oriented protocol better suited for simple message transmission. It enables the sending of datagrams without establishing a prior connection. UDP transmits smaller data units, making it faster and less burdensome than TCP. However, it lacks ordering, reliability, and duplication control [100].

2.7.3.6 Application Layer

The application layer contains protocols and methods that allow user interfaces and applications to communicate with the network. It serves as an abstraction layer, hiding the underlying transmission processes from the application. It relies on all lower layers to complete its functionality. The most commonly used application layer protocols include HTTP, CoAP, WebSocket, MQTT, XMPP, DDS, and AMQP.

- **HTTP:** The Hypertext Transfer Protocol is one of the most commonly used protocols in the application layer. Its primary function is to transmit hypermedia documents such as HTML. It follows a classic client-server model where a client sends a request and waits for a response. HTTP is stateless, meaning the server does not retain session information between requests [104].

- **CoAP:** The Constrained Application Protocol is designed for Machine-to-Machine (M2M) communication and is intended for use in constrained networks such as low-power and lossy networks. It operates over UDP rather than TCP, which helps reduce bandwidth usage. Features include header compression, resource discovery, asynchronous messaging, auto-configuration, congestion control, and multicast support [105].

- **WebSocket:** The WebSocket protocol enables full-duplex communication between a client (typically in a browser) and a remote host. It begins with an initial "handshake" and follows with a message-based data exchange over TCP. WebSocket is ideal for browser-based applications requiring two-way communication without the overhead of multiple HTTP connections [104].

- **MQTT:** The Message Queuing Telemetry Transport protocol follows a publish/subscribe model. It is lightweight, open, and simple to implement, making it ideal for constrained environments like M2M and IoT, where minimal code footprint and low bandwidth usage are essential. It runs over TCP/IP or other reliable network protocols [106].

- **XMPP:** The Extensible Messaging and Presence Protocol supports near-real-time transmission of XML data between network entities. It has broad applications, including instant messaging (e.g., Google Talk). Despite its wide adoption, XMPP is relatively complex and may be too resource-intensive for constrained IoT devices [107].

- **DDS:** The Data Distribution Service is a middleware protocol designed for device-to-device or M2M communication. It follows a publish/subscribe model, offering configurable quality-of-service settings and reliability guarantees [92].

- **AMQP:** The Advanced Message Queuing Protocol is an open standard developed primarily for financial messaging. It supports both point-to-point and publish/subscribe models. AMQP relies on a message broker to route data from producers to consumers, allowing asynchronous communication and message queuing [100].

2.7.4 Technologies in Middleware Layer

2.7.4.1 Cloud Computing

Cloud computing is an internet-based development model ("cloud") and the use of computing technologies ("computing"). It is a general term for everything that involves the delivery of hosted services over the internet. It is used to describe both a platform and a type of application. These cloud applications use large data centers and powerful servers to host web applications and web services. Anyone with a proper internet connection and a standard browser can access a cloud application. Therefore, cloud-centered IoT architectures provide computing and storage resources for all the data collected by IoT devices, which can later be subjected to analysis or data mining techniques for information retrieval and knowledge discovery from the collected data [95].

Below are the different types of services that can be offered through cloud computing:

- **Infrastructure as a Service (IaaS):** Provides virtual servers with unique IP addresses and on-demand storage blocks. Clients pay exactly for the amount of service they use, similar to utilities like electricity or water—this service is also called utility computing. IaaS is the delivery of hardware (server, storage, and network) and associated software (virtualization technology, operating systems, file systems) as a service. It is an evolution of traditional hosting that does not require long-term commitments and allows users to provision resources on demand

[92]. Amazon Elastic Compute Cloud (EC2) and Secure Storage Service (S3) are examples of IaaS.

- **Platform as a Service (PaaS):** PaaS is a set of software and development tools hosted on the provider's servers. Google Apps is one of the most well-known PaaS providers. The idea is that a provider offers both hardware (as in IaaS) and a certain amount of application software, such as integration into a common set of programming functions or databases, as a base upon which applications can be built. PaaS is a platform for application development and deployment delivered as a service to developers over the internet. It simplifies the development and deployment process without the cost and complexity of buying and managing a complete infrastructure, offering all the facilities needed to support the full lifecycle of creating and deploying applications and web services [108].

- **Software as a Service (SaaS):** In SaaS, the provider allows the customer to use its applications. The software interacts with the user through an interface. These applications can range from email to platforms like Twitter. The idea is that a provider offers a hosted software package (running on a platform and infrastructure) that is not owned by the user but is paid for based on some usage or consumption metrics. No development or programming is required, although users may need to configure the software (which is often flexible, configurable, and sometimes customizable). Nothing needs to be purchased—users only pay for what they use. A SaaS provider typically hosts and manages a given application in its own data center and makes it available to multiple tenants and users over the network. Some SaaS providers run on top of other providers' PaaS or IaaS services. Oracle CRM on Demand, Salesforce.com, and Netsuite are some well-known examples of SaaS [108].

Chapter 3

System Design

This chapter provides a comprehensive overview of the system's structure and functionality, with a primary focus on the requirements analysis phase within the waterfall methodology. This phase serves as a fundamental precursor to system development, aiming to elucidate user needs, constraints, and functionalities. By emphasizing the importance of meticulous requirement comprehension and documentation. Moreover, as the chapter navigates through the landscape of the IoT NILM system, it contextualizes the system's purpose and role within this burgeoning domain, shedding light on its unique value proposition and potential impact within the broader IoT ecosystem.

3.1 Requirements and Purpose Specifications

- **System's purpose:** Use smart metering technology and machine learning algorithms to automatically detect and classify appliance usage events in real-time, enabling efficient energy management, proactive maintenance, behavioral insights, and grid optimization.

- **Behavior:** The system consists of two main elements: the smart meter device and the cloud system. The smart meter generates a new sample of phase frequency, total and fundamental RMS current and voltage, active, reactive, and apparent power, power factor, and THD at a fixed rate of 10 ms. This information is then used to feed an event detection algorithm embedded in the smart meter and is also instantly transmitted to the cloud. If an on/off event has occurred, the RMS current and active/reactive power transient are also sent to the cloud.
  Within the cloud, three main components are deployed: the user interface, the

database, and the machine learning algorithms. The user interface is utilized to display consumption information to the user, while the database stores the history of all electrical parameters. The machine learning algorithms consist of a clustering algorithm that automatically creates a dataset and neural networks to extract features from appliance signatures and classify appliances.

- **System Management Requirements:** A user interface to visualize data in real-time and apply control to specific functions of the system.
- **Data Analysis Requirements:** Data processing is performed both within the smart meter and in the cloud. Electrical parameters and event detection algorithms are embedded in the smart meter circuit, while all machine learning algorithms are deployed in the cloud.
- **Application Deployment Requirements:** Deployment of the web interface on a web server within a cloud computing platform as a service.
- **Security Requirements:** User authentication services.

3.2 Process Specification

Previously, it was mentioned the functionality of the proposed system, and it was outlined that it consists of two main components: the smart meter and the cloud system. So to detail the process specification both elements have to be analyzed separately; the flowchart of the process for the smart meter and the cloud system can be seen in Figures 3.1 and 3.2.

The flowchart in Figure 3.1 begins with the initialization of the ADE9000, followed by checking for a successful WiFi connection. If the WiFi connection fails, the process loops back to the WiFi setup. If the WiFi connection is successful, the next step is to establish a connection with an MQTT broker. Failure to connect to the MQTT broker also loops back to reattempting the MQTT broker connection. However, if both the WiFi or broker connections remain unsuccessful after 10 seconds, the smart meter switches to an offline mode. In this mode, the device operates exactly as if it were online, but all the collected data is stored locally on an SD card instead of being transmitted. Once both connections are successfully established, the system proceeds to gather a series of measurements, including line frequency, voltage and current RMS, active, reactive and apparent power, energy consumption in kilowatt-hours and in kilovar-hours, and other related parameters.

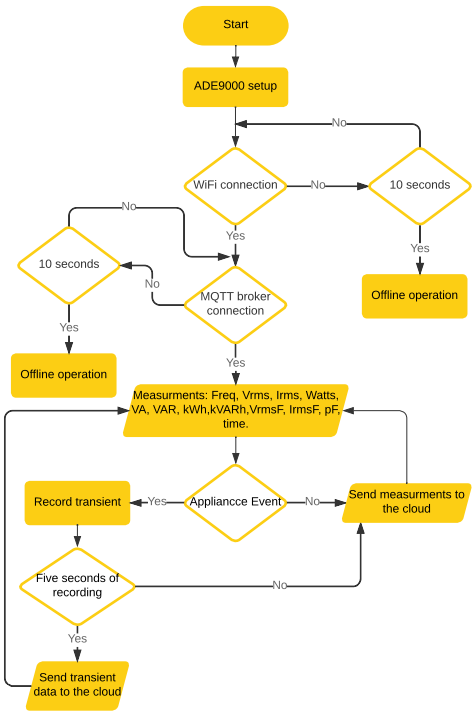


Figure 3.1: Flowchart of how the smart meter works.

Following the measurement collection, the flowchart checks for the occurrence of an appliance event. If an appliance event is detected, the system records transient data for five seconds. After the recording period, the transient data is sent to the cloud. If no appliance event is detected, the system sends the regular measurements to the cloud. This process ensures that both regular operational data and specific event-related data are captured and transmitted for further analysis.

Figure 3.2 illustrates how measurements and appliance events are processed within the web server. This flowchart does not include the process of how data is displayed in real time on the web app, as the web application receives data directly from the broker through MQTT with WebSockets.

The process begins with user authentication, including login and permission verification. Once the user is authenticated and permissions are granted, the system establishes a connection to an MQTT broker. If any of these steps fail, the system loops back to retry the respective step. Upon successful connection, the system reads data from topics in the MQTT broker.

The next step involves detecting new appliance events. If no new events are detected, the system sends the collected measurements to an Atlas database. If a new appliance event is detected, the process checks if a dataset has been previously created. If a dataset exists and the Deep Learning (DL) model is already trained, the event is classified, the

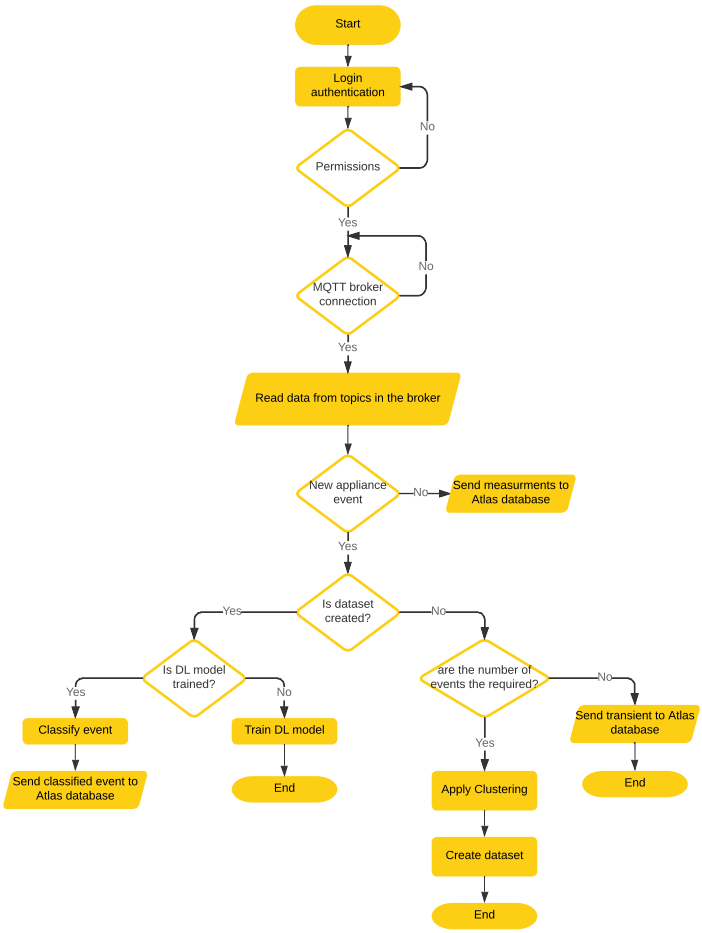


Figure 3.2: Flowchart of how measurements and events are treated inside the web server.

classified event is sent to the Atlas database, and the whole process finishes. If the DL model is not trained, the system proceeds to train the model. If no dataset is available, the system checks if the required number of events has been reached. If this is the case, clustering is applied, a dataset is created, and the process ends. If the required number of events is not met, the transient data is sent to the Atlas database, and the process concludes.

3.3 Domain Model Specification

The domain model specification provides a structured representation of the main concepts, entities, and objects within the IoT system. This model defines the attributes of each object and their interrelationships, creating an abstract layer that guides the system's design. For this IoT system, the primary goal of the domain model is to facilitate the accurate detection and classification of appliance usage events, ensuring efficient data flow from physical measurements to cloud-based processing. The overall

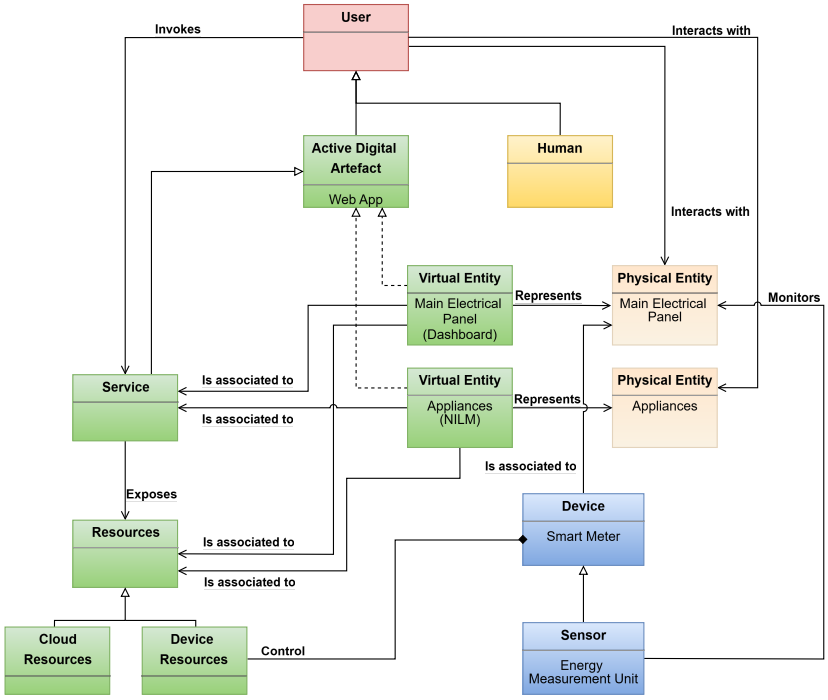


Figure 3.3: Domain model diagram representing the main concepts, entities, and interrelationships within the NILM IoT system.

structure of this model is illustrated in Figure 3.3.

3.3.1 Physical Entity

A Physical Entity refers to any identifiable object in the physical environment that the IoT system monitors or controls. In the context of this smart metering IoT system, Physical Entities are the appliances that the system detects, classifies, and monitors, as well as the main electrical panel that serves as the source of electricity for the residence.

- Main Electrical Panel: The main electrical panel serves as a central Physical Entity, acting as the distribution point for electricity within the residence. The smart meter device is installed at this panel to gather various electrical parameters (e.g., phase frequency, RMS current and voltage, active, reactive, and apparent power, power factor, and Total Harmonic Distortion).

- Appliances: Appliances, such as refrigerators, washing machines, and air conditioners, represent the other Physical Entities monitored by the system. These entities are not directly connected to the IoT system; instead, their operation is inferred through variations in electrical parameters, enabling detection and classification.

3.3.2 Virtual Entity

For each Physical Entity, there is a corresponding Virtual Entity in the cloud. Virtual Entities represent the Physical Entities in the digital domain, allowing the IoT system to analyze and store usage patterns, operational states, and other relevant data.

- Virtual Electrical Panel: The Virtual Electrical Panel is a digital representation of the main electrical panel in the cloud, implemented as a dashboard interface. It provides a consolidated view of real-time and historical electrical measurements, detected events, and energy distribution patterns collected by the smart meter device.

- Virtual Appliances: virtual Appliances represent each Physical Appliance monitored by the system. These entities are inferred using machine learning algorithms, which classify appliances based on the electrical signatures identified by the smart meter. The Virtual Appliance entities enable data-driven insights, including appliance-specific energy consumption and usage behavior.

3.3.3 Devices

The Device serves as the intermediary layer between Physical Entities and Virtual Entities, facilitating data collection and actuation. In this system, the Smart Meter is the primary device responsible for gathering sensor data and transmitting it to the cloud.

- Smart Meter Device: The smart meter is the main Device within the system, equipped with sensors that collect real-time measurements of various electrical parameters. It includes WiFi connectivity for data transmission and contains embedded algorithms for on-device event detection. Once an event is detected, the device sends both regular measurements and transient data to the cloud, providing the foundation for appliance classification and real-time monitoring.

3.3.4 Resources

Resources are software components that provide specific functions within the system. Resources can be located either on the smart meter device (on-device resources) or in the cloud (network resources).

- On-Device Resource: The primary on-device resource is the event detection algorithm embedded within the smart meter. This algorithm monitors real-time

data and detects on/off events based on predefined conditions, triggering the transmission of transient data to the cloud whenever an event occurs.

- Network Resource: Network resources are hosted in the cloud and include components such as the database, machine learning algorithms, and user interface. The cloud database stores historical data on electrical parameters and events, while machine learning algorithms perform clustering and classification of appliances. The user interface resources are deployed as part of the web application, allowing end-users to view appliance usage data and other insights.

3.3.5 Services

Services provide interfaces that allow users and system components to interact with Physical Entities, accessing the necessary data and functionalities for the IoT system's operation. Services can retrieve information about the status of appliances or perform actions based on user inputs.

- Data Visualization Service: This service provides a user interface for visualizing real-time and historical data, offering users insights into their energy consumption and appliance usage patterns. This service is implemented as a web application that directly connects to the cloud database and displays relevant information to the user.

- Appliance Classification Service: Leveraging machine learning algorithms, this service processes transient data to classify appliances based on their electrical signatures. This classification enables the system to identify specific appliances and generate usage insights, contributing to efficient energy management.

- Data Storage Service: The Data Storage Service manages the cloud database, which stores real-time and historical measurements, detected events, and appliance classifications. This service ensures that data is organized, accessible, and available for analysis by other system components.

3.4 IoT Level Specification

The proposed NILM system aligns with Level 3 of the IoT hierarchy [92], which centralizes data storage, analysis, and application functionality in the cloud, while the on-site device primarily performs data collection, as shown in Figure 3.4. This setup

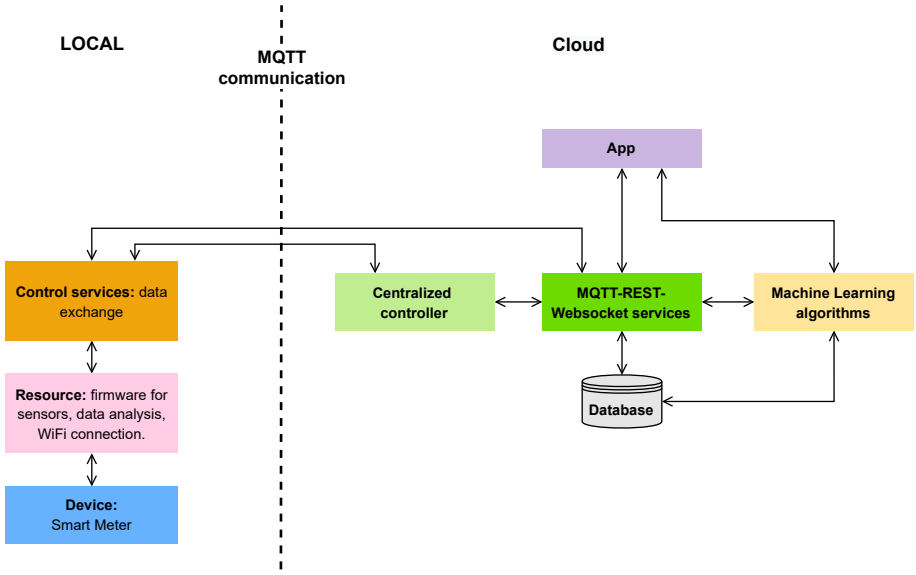


Figure 3.4: IoT description of the NILM system.

allows the NILM system to leverage cloud resources for computationally demanding tasks, while a single, central smart meter serves as the primary data source.

3.4.1 Single-Node Architecture with Cloud-Centric Analysis

The NILM system consists of two primary components: a central smart meter and a cloud-based backend system. The smart meter, acting as a centralized data collector, measures comprehensive electrical parameters such as frequency, RMS current and voltage, active, reactive, and apparent power, power factor, and Total Harmonic Distortion (THD). While preliminary event detection occurs on the meter, the device predominantly functions as a data acquisition point, forwarding raw and processed data to the cloud. Upon data collection, the cloud system manages:

- Data Storage: All electrical parameters and event data are stored in the cloud, creating a historical database for appliance usage insights and load monitoring.
- Machine Learning Analysis: The cloud hosts machine learning algorithms designed to disaggregate and classify appliance usage patterns based on data from the central meter. These algorithms enable appliance-level monitoring without requiring individual sensors on each device.
- User Interface and Reporting: A web interface, deployed in the cloud, provides end users with access to real-time appliance usage and energy consumption insights.

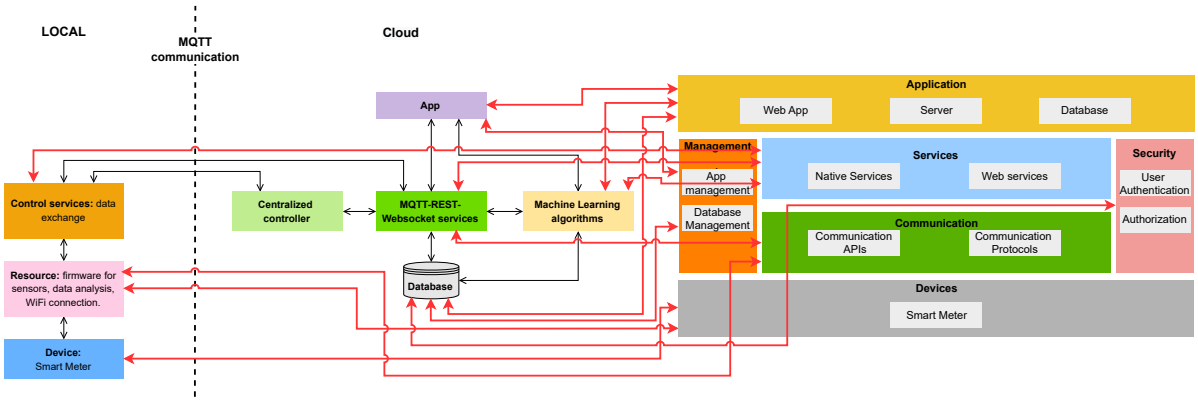


Figure 3.5: Functional view specification diagram.

3.4.2 Cloud-Centric Processing and Scalability

By offloading complex processing tasks to the cloud, the NILM system achieves scalable and real-time analytics capabilities. This approach minimizes computational demands on the smart meter while maximizing system flexibility, allowing for advanced disaggregation and appliance-level event detection in real time.

Thus, this NILM system adheres to the characteristics of a Level 3 IoT system by:

- Using a central data collection node (the smart meter) as the sole physical point of measurement.
- Centralizing data processing, storage, and user access in the cloud.
- Leveraging a cloud-based user interface, where users can access real-time appliance disaggregation and historical consumption analytics.

3.5 Functional View Specification

The functions of the IoT system can be grouped based on their roles, including Devices, Communication, Services, Management, Security, and Application. The interactions between these groups and the domain model instances are outlined below and can be seen in Figure 3.5.

1. Devices: this group includes the hardware components and measurement instruments.
    - Smart Meter: Captures electrical parameters such as RMS current and voltage, active/reactive power, and total harmonic distortion (THD). It

performs ON/OFF event detection locally but does not include NILM algorithms.

2. Communication: the communication group encompasses protocols that enable data transfer between the smart meter and the server.

    - MQTT Protocol: Utilized for sending regular electrical measurements and event-related transient data from the smart meter to the server in real-time.
    - WebSocket Protocol: Supports real-time data updates in the user interface, facilitating low-latency communication.
    - REST API: Provides endpoints for data retrieval, configuration settings, and user interactions with the NILM system.

3. Services: backend services and cloud-based functionalities.

    - Measurement Service: Handles the reception of regular electrical data from the smart meter via MQTT.
    - Event Detection Service: Processes ON/OFF event data transmitted by the smart meter, triggering further analysis.
    - NILM Analysis Service: Executes machine learning algorithms on the server for appliance classification using the event and transient data received from the smart meter.

4. Management: these functions involve configuration and system control.

    - System Configuration: Allows for adjustments to the smart meter settings (e.g., sampling rate, event detection thresholds) and cloud-based NILM model parameters.
    - Data Management: Interfaces with the web application and database, organizing historical measurement data and classification results for analysis and user feedback.

5. Security: this group ensures data protection and access control.

    - User Authentication and Authorization: Ensures secure access to the web application and API endpoints, with role-based permissions.
    - Data Integrity and Encryption: Protects the transmitted data (measurement and event data) from unauthorized access, particularly during the transfer between the smart meter and server.

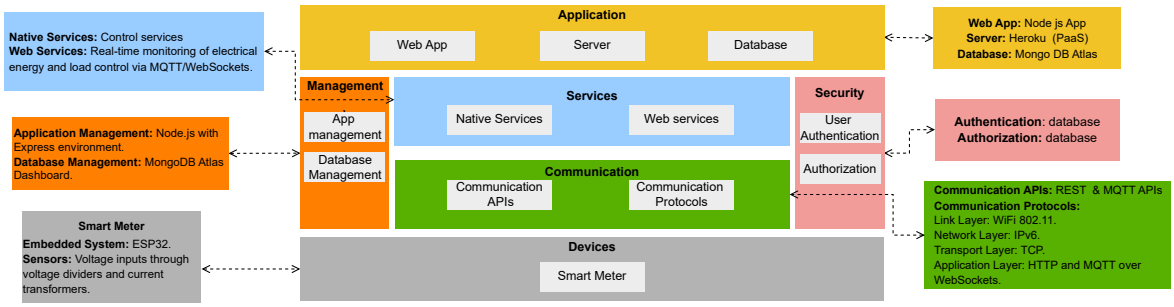


Figure 3.6: Operational view specification diagram.

6. Application: the application section connects the system to the end-user interface and backend processing:

    - Web Application: Provides an interface for real-time monitoring of electrical parameters and visualization of disaggregated appliance data.
    - NILM Dashboard: Displays appliance-level energy usage insights, disaggregation results, and historical consumption trends.
    - Database Integration: Manages the storage of electrical data and NILM classification outcomes, supporting efficient querying and analysis.

3.6 Operational View Specification

In this section the technologies used in the functional groups of the NILM IoT system are defined. These details can be observed in Figure 3.6.

- Devices:
    - Central Meter:
        * Embedded System: ESP32.
        * Sensors: Voltage inputs through voltage dividers and current transformers (split-core type).
- Communication:
    - Communication APIs: REST & MQTT.
    - Communication Protocols:
        * Link Layer: WiFi 802.11.
        * Network Layer: IPv6.

    - Transport Layer: TCP.
    - Application Layer: HTTP and MQTT over WebSockets.

- Services
  - Native Services: Control services implemented in embedded systems.
  - Web Services: NILM and real-time monitoring of electrical energy via MQTT/WebSockets.
- Management:
  - Application Management: Node.js with Express environment.
  - Database Management: MongoDB Atlas Dashboard.
- Security:
  - Authentication: Managed via database.
  - Authorization: Managed via database.
- Application:
  - Web Application: Node.js web application.
  - Server: Heroku (PaaS).
  - Database Server: MongoDB Atlas.

Chapter 4

System Implementation

This chapter details the practical development of the proposed NILM system, focusing on the integration of hardware and software components that enable real-time operation. The implementation follows the architectural framework structured around four key stages: data acquisition, event detection, feature extraction, and load classification. Emphasis is placed on building a modular and scalable system suitable for residential environments.

The following sections describe the custom-designed smart meter, the algorithms used for real-time event detection and feature extraction, and the generation of an adaptive training dataset using unsupervised learning. Additionally, the implementation of supervised classification models and the development of a web-based platform for data visualization and user interaction are presented, highlighting the end-to-end functionality of the system.

4.1 NILM Architecture

As outlined in Section 1.5, the proposed NILM system adopts an event-based methodology to support real-time operation. The overall structure comprises four key stages: data acquisition, event detection, feature extraction, and load recognition. Each stage is essential for improving the accuracy and reliability of the NILM process. Based on these steps, the system architecture is illustrated in Figure 4.1. The architecture includes a smart meter that captures household energy data, including appliance-level signatures. This data is transmitted to the cloud, where it is visualized through a user interface and stored in a database. Clustering techniques are employed to derive a labeled dataset from the initially unlabeled data. This dataset is then used to train a deep learning model intended for deployment in the cloud. Since the labels in clustering

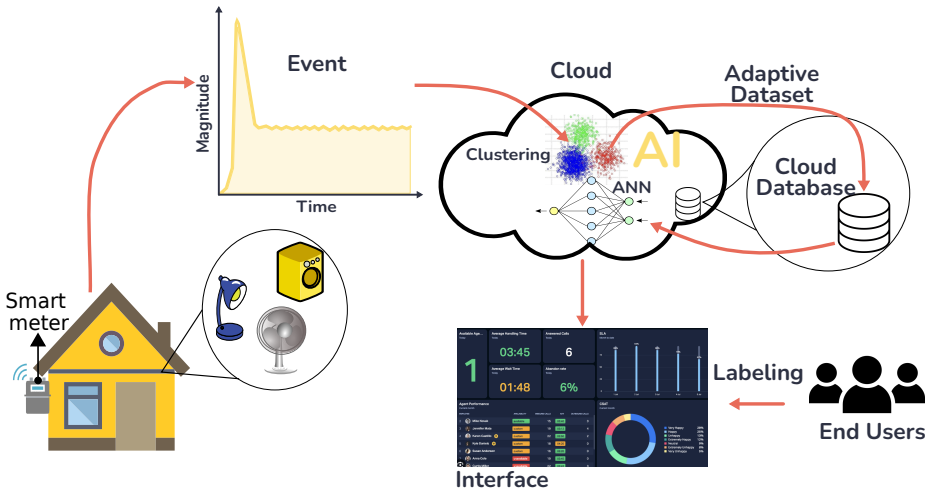


Figure 4.1: Proposed system architecture.

are generated numerically, the final assignment of appliance names would be a task of the end user once the system is fully operational.

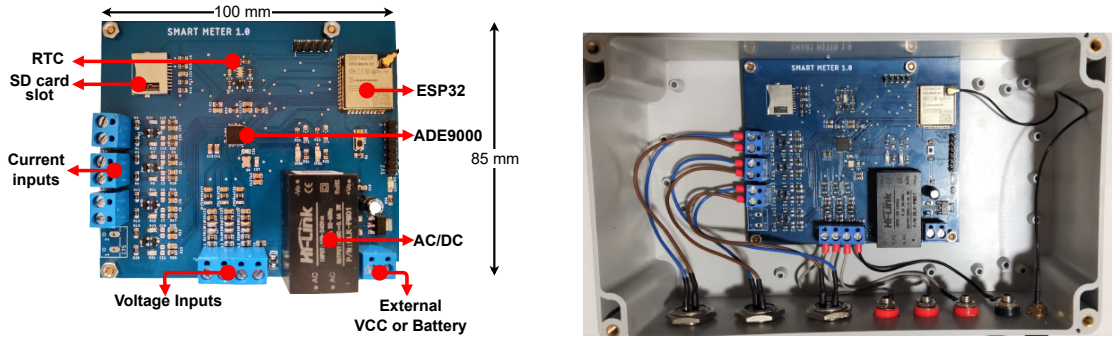
4.2 Data Acquisition Device

The quality of data is a critical factor in the performance of any AI-based application, particularly in the context of NILM systems. To ensure high data availability and flexibility, a custom three-phase IoT smart meter was developed for this work. Its physical structure and labeled components are shown in Figure 4.2. This smart meter is designed for installation in the main electrical panel of a residential building, where it continuously measures electrical parameters and wirelessly transmits them to the cloud in real time.

Unlike conventional smart meters that primarily report cumulative active energy consumption, the proposed device offers a comprehensive set of electrical parameters. These include total and fundamental RMS current and voltage, active, reactive, and apparent power, reactive energy, power factor, and a variety of power quality indicators such as line frequency, the amplitude of 64 current harmonics, and total harmonic distortion (THD). This enriched dataset enhances the potential for precise load identification and classification.

As illustrated in Figure 4.2, the smart meter integrates several key components, each serving a specific function within the system:

- **ESP32**: Acts as the central microcontroller, responsible for wireless communication and local processing.



(a) PCB and components of the SM. (b) Connections in ABS abs enclosure box.

Figure 4.2: Printed circuit board and enclosure.

- **ADE9000**: An advanced energy metering IC that performs real-time measurement of electrical quantities and power quality metrics.
- **RTC (Real-Time Clock)**: Maintains accurate timekeeping to enable precise timestamping of measurements.
- **SD Card Slot**: Provides local storage capability for logging data in scenarios where internet connectivity is limited or unavailable.
- **Voltage and Current Inputs**: Terminal blocks used to connect the device to the electrical system and current transformers.
- **External VCC or Battery Input**: Offers an alternative power source, allowing battery backup or flexible deployment in off-grid settings.

One of the most important characteristics of data acquisition systems is the sampling rate. In this implementation, the SM does not output raw high-frequency waveform data but instead provides averaged electrical measurements every 10 milliseconds. While this excludes detailed transient information, it significantly reduces the data bandwidth required for transmission and storage, making the system suitable for real-time, scalable NILM applications.

Moreover, the SM incorporates an onboard event detection algorithm, which is essential for identifying appliance state transitions. This algorithm and its role within the broader NILM framework are described in the following section.

4.3 Event detection algorithm

Event detection algorithms in time series data are essential for identifying significant occurrences or patterns within temporal sequences. There are numerous techniques

documented in the literature proposed for this purpose, including the CUSUM algorithm and its variations, Bayesian change point detection, and the PELT method [109, 110, 111]. In this study, a real-time event detection system inspired by the CUSUM algorithm was integrated into the SM. The summarized steps can be observed in Figure 4.3.

1. Sliding window means: The RMS current feature is used to find new appliance events. When the meter starts running for the first time, it creates a window consisting of 'n' elements. Once the window is full, the new RMS current value enters the array, while the oldest value shifts out, enabling real-time analysis.

2. Deviation: To enhance the robustness of event detection, instead of calculating the deviation between the new measurement (x_{new}) and the previous one (x_{new-1}), the deviation is computed using the sliding window mean ($\bar{x}$), as described in Equation 4.1.

$$\Delta = x_{\text{new}} - \overline{x} \tag{4.1}$$

3. Threshold: A fixed threshold of 300 mA RMS was selected for event detection due to the limitations of adaptive thresholds, such as those based on mean or standard deviation, where sensitivity increases significantly in stable input signals. The value of 300 mA was arbitrarily chosen, since in a residential context appliances drawing lower currents generally represent minor loads and have a negligible contribution to the overall energy consumption. An appliance could trigger a rising (ON) or falling (OFF) event, the nature of which is computed using the following equations:

$$On_{event} = \Delta > Threshold, \tag{4.2}$$

$$Off_{event} = \Delta < -Threshold. \tag{4.3}$$

4. Event selection: To accurately discern whether an event corresponds to an appliance switching ON/OFF status, several conditions were taken into account. For instance, if an appliance exhibits a peak during its transient phase, the subsequent decline of the peak could have been labeled as a false falling event. False rising/falling events or events lasting less than 5 seconds are disregarded. While testing the algorithm, a 5-second transient signal proved to be long enough to describe all types of loads, especially those with resistive parts whose transients can last up to a few seconds, as demonstrated in Chapter 5 and also discussed in [112].

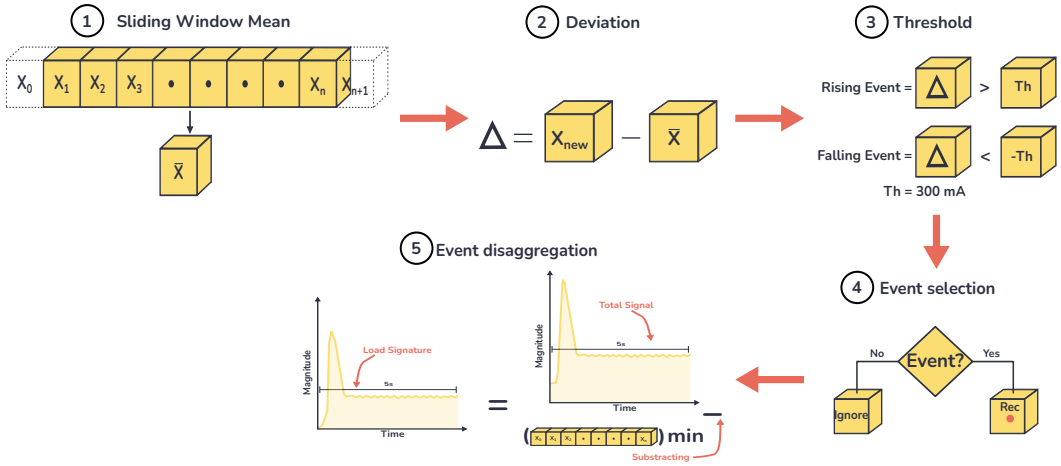


Figure 4.3: Event detection process.

5. Event disaggregation: Since the SM is installed on the main panel, it measures the aggregate of all residential loads. To isolate each appliance's signature, the baseline of the event must be properly removed. For this purpose, the smallest value within the sliding window, $\min(S[0 : \text{windowLength} - 1])$, is subtracted from the entire 5-second event signal $S = [s_1, s_2, ..., s_n]$. This operation can be expressed as:

$$\text{disaggregation}[i] = s_i - \min(S[0 : windowLength - 1]). \tag{4.4}$$

This subtraction ensures that the event signal is aligned to a zero reference, avoiding the influence of background consumption or measurement offsets. Using the minimum value as the baseline is particularly effective because it captures the lowest steady level observed during the transient window, thereby enabling a more accurate extraction of the appliance's distinctive signature. The algorithm primarily works with RMS current, but the final disaggregated signatures include RMS current, active power, and reactive power.

4.4 Automatic and Adaptive Dataset

The NILM field frequently relies on public datasets or manually crafted fixed datasets for research and application purposes. This work presents the creation of a dataset tailored to the end user's requirements, with the algorithms having no prior knowledge of the types or quantities of appliances in the household. Consequently, an unsupervised learning approach is employed. Basically, when an event is detected, the signatures are

sent to the Atlas MongoDB web database, but the dataset building process starts after "n" events have occurred. Once there are "n" events available, the clustering algorithm creates groups of events that correspond to the ON/OFF signatures of different loads. These clusters are not labeled with the true names of the appliances; instead, they are just numbered. The correct names will be set by the end users when the dataset is already formed and working.

4.4.1 Feature Extraction with 1-D CNN Autoencoders

Feature extraction involves identifying and selecting relevant information from raw data to improve model performance. Techniques include statistical analysis, filtering, and deep learning-based feature learning [113]. As unsupervised learning is required in this work, a 1D Convolutional Neural Network (1D-CNN) autoencoder was selected for feature extraction. This architecture compresses input data into a latent representation through convolutional layers. The encoder reduces dimensionality by capturing patterns in the data. Then, the decoder reconstructs the original input from this compressed representation using transposed convolutional layers. During training, the model learns to minimize the difference between the input and output, optimizing the convolutional filters to capture salient features. The training objective can be formalized as minimizing the Mean Squared Error (MSE) between the input x and the reconstructed output $\hat{x}$, which is given by Eq. 4.5, where $\mathcal{L}(x, \hat{x})$ represents the loss function.

$$\mathcal{L}(x, \hat{x}) = \frac{1}{n} \sum_{i=1}^{n} (x_i - \hat{x}_i)^2 \tag{4.5}$$

Through this iterative process, the autoencoder discovers a compressed representation of the input data while retaining essential information. This latent representation is what can be used as features [114]. The designed 1-D CNN Autoencoder architecture can be found in Figure 4.4.

4.4.2 Clustering with Automatic k Centroids

To generate an adaptive dataset, a clustering approach was integrated into the system following the feature extraction stage. Although the K-means algorithm had been previously described (see Section 2.6.1), its application here plays a critical role in organizing the high-dimensional feature vectors into meaningful groups corresponding to different appliance signatures. Since K-means requires the number of clusters k to be predefined, the system includes a mechanism to determine the optimal k automatically.

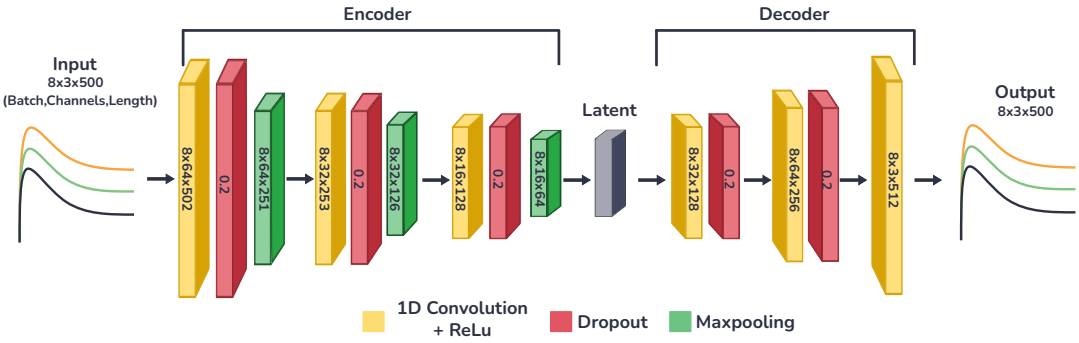


Figure 4.4: 1-D CNN autoencoder architecture

This is achieved by evaluating silhouette scores over a range of candidate values. For each tested k, clustering is performed and the average silhouette score is computed, quantifying the cohesion and separation of the resulting clusters. The silhouette score for a data point i is calculated as:

$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i))}. \tag{4.6}$$

In this equation, $a(i)$ represents the average intra-cluster distance (i.e., the average distance from point i to all other points in the same cluster), and $b(i)$ denotes the smallest average distance from i to points in a different cluster. The value of k that yields the highest average silhouette score is selected as the optimal number of appliance classes for the given household context.

This unsupervised grouping enables the system to generate a self-adaptive dataset, where each cluster can later be labeled by the user, specifying which device each group corresponds to. This approach eliminates the need for manually crafting data in advance, making the system more scalable and adaptable to different environments.

4.5 Supervised Classification Models

Once the labeled dataset is available, an ANN is trained to classify new events generated by appliances in real time. As mentioned previously, various models are suitable for time series data, such as 1D CNN and LSTM, each offering specific advantages depending on the signal characteristics and application context.

To determine the most appropriate architecture for the NILM system, three different models were implemented and evaluated. The details of each model are presented in the following subsections, while the selection criteria and performance analysis are discussed in Chapter 5.

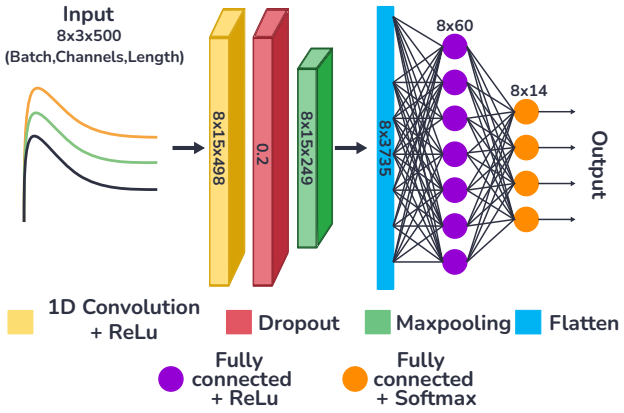


Figure 4.5: 1-D CNN architecture for classification.

4.5.1 1D-CNN Implementation

The 1D-CNN model was built using the PyTorch framework. The model processes input tensors of size $8 \times 3 \times 500$, where 8 is the batch size, 3 represents the number of channels (corresponding to I_{RMS}, P, and Q), and 500 is the length of the time window. The architecture is illustrated in Figure 4.5 and consists of the following components:

- **1D Convolution + ReLU**: A convolutional layer with 15 output channels and a kernel size that reduces the temporal dimension from 500 to 498, followed by a ReLU activation function.
- **Dropout**: A dropout layer with a rate of 0.2 is applied to prevent overfitting.
- **MaxPooling**: A max-pooling operation with a stride of 2 halves the temporal dimension, resulting in a tensor of size $8 \times 15 \times 249$.
- **Flatten**: The output is flattened into a vector of size 8×3735 to serve as input to the fully connected layers.
- **Fully Connected + ReLU**: A dense layer with 60 ReLU-activated neurons transforms the flattened vector into a latent representation.
- **Output Layer + Softmax**: A final dense layer with 14 neurons applies the softmax function to produce class probabilities, with each neuron corresponding to an appliance label.

The model's hyperparameters—including kernel size, dropout rate, number of channels, and number of fully connected units—were tuned using PyTorch's integrated grid search strategy to achieve optimal classification performance.

4.5.2 WaveNet Model

The WaveNet model, originally designed for speech synthesis, can be highly effective for time series classification, as it is able to capture long-range dependencies in sequential data. In the context of this project, where the inputs consist of I_{RMS}, P, and Q waveforms from detected appliance events, the WaveNet model operates as follows:

- **Causal Convolutions**: The first operation applied to the input is a 1D causal convolution layer of kernel size 1×3, which outputs a feature map of shape $8 \times 90 \times 500$. This layer ensures that each output at time step t is influenced only by inputs from time steps $\leq t$, preserving the temporal order of the signal. The causal convolution operation is defined as:

$$y_t = \sum_{k=0}^{K-1} w_k x_{t-k}, \tag{4.7}$$

  where y_t is the output, x_{t-k} the input, and w_k the weights of the filter.

- **Residual Blocks with Dilated Convolutions**: The output then passes sequentially through five residual blocks, where each block uses the residual output of the previous one to compute its own output. Each block contains a gated activation unit consisting of a dilated convolution of kernel size 1×3, followed by two parallel activation functions: tanh and σ. These two branches are then multiplied element-wise. The result is passed through a 1×1 causal convolution to produce both the residual output (which feeds into the next block) and a skip connection. The use of increasing dilation rates in each block allows the receptive field to grow exponentially. The dilated convolution is defined as:

$$y_t = \sum_{k=0}^{K-1} w_k x_{t-dk}, \tag{4.8}$$

  where d is the dilation factor that increases with each layer to capture broader temporal patterns efficiently.

- **Skip Connections and Aggregation**: The skip connections from each residual block are aggregated through summation, producing a feature map of shape $8 \times 15 \times 500$. This mechanism helps preserve gradients and facilitates training of deeper models by allowing information to flow directly to the output.

- **Flatten and Dense Layers**: The aggregated features are flattened into shape 8×7500 and passed through two fully connected layers. The first contains 120

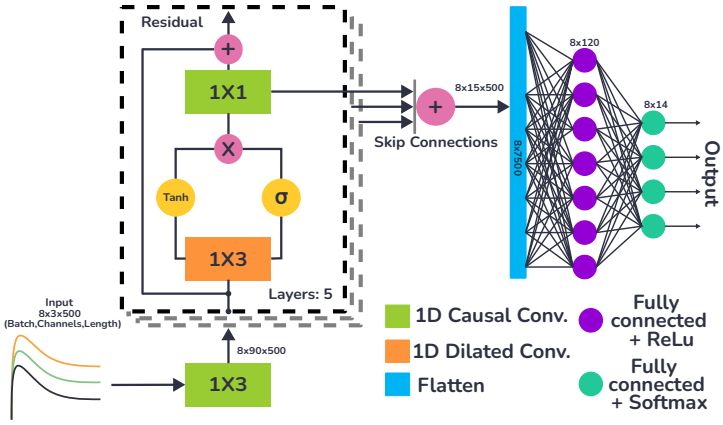


Figure 4.6: WaveNet model used for classification tasks.

neurons with ReLU activation, and the second uses 14 neurons with softmax activation to output the class probabilities.

Similar to the 1D-CNN model, the WaveNet model was implemented using the PyTorch framework and the grid search technique was employed to determine the optimal hyperparameters for training. The complete architecture used in this work, which includes causal convolutions, stacked residual blocks with dilated convolutions, and a final classification layer, is depicted in Figure 4.6.

4.5.3 LSTM Model

The third architecture explored for appliance classification is based on a Long Short-Term Memory network, which is well-suited for time series tasks due to its ability to retain information over long sequences. The model combines convolutional and recurrent layers to enhance feature extraction and temporal dependency modeling. The architecture, as illustrated in Figure 4.7, processes the input time series of shape $8 \times 3 \times 500$ (batch size, channels, and time steps). The structure operates as follows:

- **1D Convolution + ReLU**: A one-dimensional convolutional layer with ReLU activation maps the three input channels to 90 output channels using a kernel size of 5, resulting in an intermediate tensor of shape $8 \times 90 \times 496$.
- **Dropout Layer**: A dropout layer with a probability of 0.3 is applied to reduce overfitting and improve generalization.
- **MaxPooling**: A max-pooling operation with a kernel size of 2 reduces the temporal dimension by half, producing a tensor of shape $8 \times 90 \times 248$.

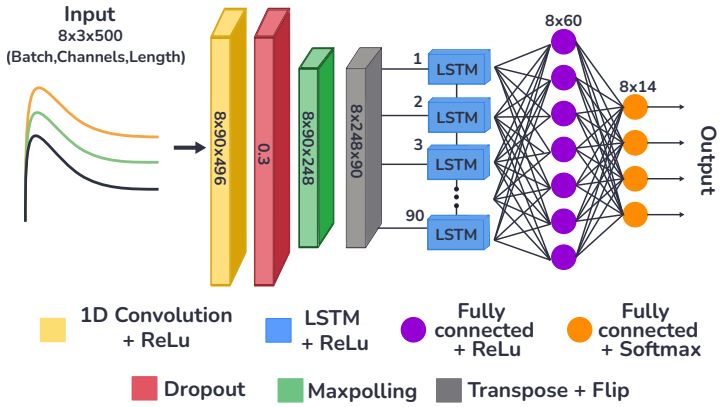


Figure 4.7: LSTM-based model used for appliance classification.

- **Transpose and Flip**: The tensor is transposed to $8 \times 248 \times 90$ and then flipped along the time axis to maintain the correct temporal order for the LSTM input.
- **LSTM + ReLU**: The output is passed through a single-layer LSTM with 90 hidden units. Only the last time step's output is used for further processing. A ReLU activation follows the LSTM layer.
- **Fully Connected Layers**: A fully connected layer with 60 units and ReLU activation maps the LSTM output to a latent representation. Finally, another dense layer with 14 output units and a softmax activation performs the classification into appliance labels.

This hybrid architecture leverages both spatial feature extraction and sequence modeling, which is advantageous in NILM applications. The complete model layout is shown in Figure 4.7.

4.6 Web Application

4.6.1 Backend

The backend of the NILM system was developed using Node.js and the Express framework, serving as the core of the cloud-based infrastructure. It is responsible for handling real-time data streams, processing appliance event signatures, storing electrical parameters, and interfacing with the classification model.

The system receives data over the MQTT protocol using TCP/IP transport. The smart meter publishes real-time measurements and event-based waveforms to predefined MQTT topics. Upon receiving this data, the backend parses and validates

the messages, converts timestamp formats, and saves the results to a MongoDB Atlas database.

- **MQTT Broker Connection:** The backend connects to a public MQTT broker hosted by EMQX (`broker.emqx.io`). It subscribes to multiple topics including `irmsEvent`, `wattEvent`, `varEvent`, and others such as `sumPower` and `centralPhA`. These topics carry real-time waveform signatures and aggregated electrical parameters published by the smart meter.
- **Data Parsing and Timestamping:** Upon message reception, the JSON data is parsed and timestamps are formatted into UTC format using the `moment.js` library. This ensures chronological consistency in the database.
- **MongoDB Integration:** The backend uses Mongoose schemas to store data such as RMS current, active/reactive power, frequency, harmonic distortion, and event waveforms. Collections are structured for different data streams (e.g., `Pha`, `Phb`, `Phc`, `energy`, `events`, and `sumPower`).
- **REST API:** An HTTP REST API is provided for the frontend to request processed data, query historical information, and interact with real-time classifications. All API endpoints are served through Express.
- **Model Inference:** The backend also includes integration with the ONNX Runtime library, allowing the deployment of trained neural network models for appliance classification as a microservice. Incoming waveform events can be passed to the model for real-time prediction.

The backend design promotes modularity and scalability, with clear separation between the data ingestion pipeline (MQTT), the persistence layer (MongoDB), and the RESTful interface for the frontend. A high-level overview of the backend architecture is shown in Figure 4.8.

4.6.2 Frontend Interface

The frontend of the web application was developed using HTML, CSS, JavaScript, and Bootstrap. It provides users with a real-time interface that receives data from the smart meter through MQTT over WebSockets ensuring dynamic and responsive updates to the visualizations. There are multiple tabs within the interface, the two most important being **Main Dashboard** and **NILM**, shown in Fig 4.9 and 4.10. There are a number of components to the user interface, including:

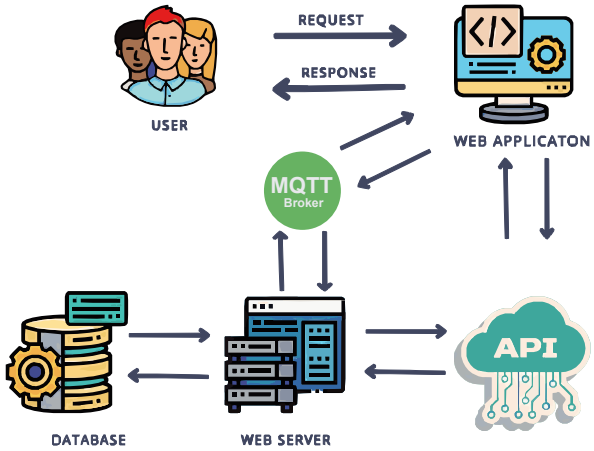


Figure 4.8: Backend architecture of the NILM system.

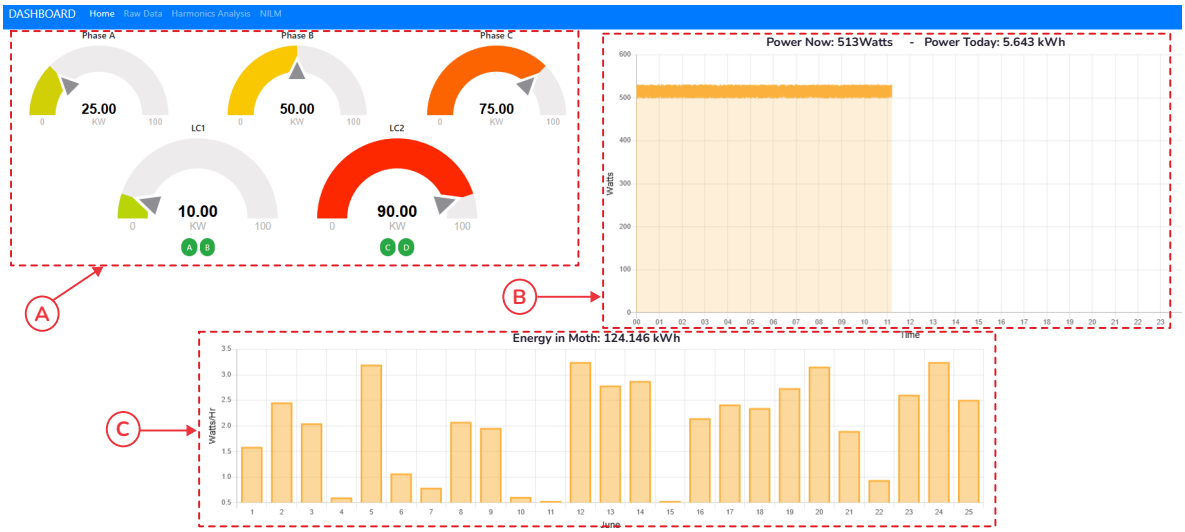


Figure 4.9: Main Dashboard tab showing real-time monitoring components: View A (gauge meters for kWs per phase), View B (area chart of daily active power), and View C (bar chart of monthly energy consumption).

- **View A**: This view displays the real-time active power consumption for each phase of the smart meter, along with any independent appliance being monitored via smart plugs. Gauge-style meters are used to intuitively convey the consumption values in kilowatts (kW).

- **View B**: A real-time area chart shows the evolution of the active power (in Watts) consumed throughout the current day. The database is checked every 5 seconds and the chart updates new values.

- **View C**: This view features a bar chart illustrating daily energy consumption (in kWh) for the current month. It provides users with a clear overview of their cumulative energy use over time.

- **View D**: Located in the NILM tab, this plot shows the waveform of the RMS current captured when an appliance event is detected and disaggregated by the smart meter. It helps to visualize the electrical signature of the event.
- **View E**: Similar to view D, this component displays the disaggregated waveform of active power over time for the detected event.
- **View F**: This view shows the reactive power waveform associated with the appliance event, completing the triad of electrical parameters.
- **View G**: This column of the NILM table lists the appliances recognized by the system.
- **View H**: Indicates the real-time status of each appliance (ON/OFF), as inferred by the disaggregation algorithm.
- **View I**: Displays the last recorded activation time of each appliance.
- **View J**: Shows the last recorded deactivation time of each appliance.

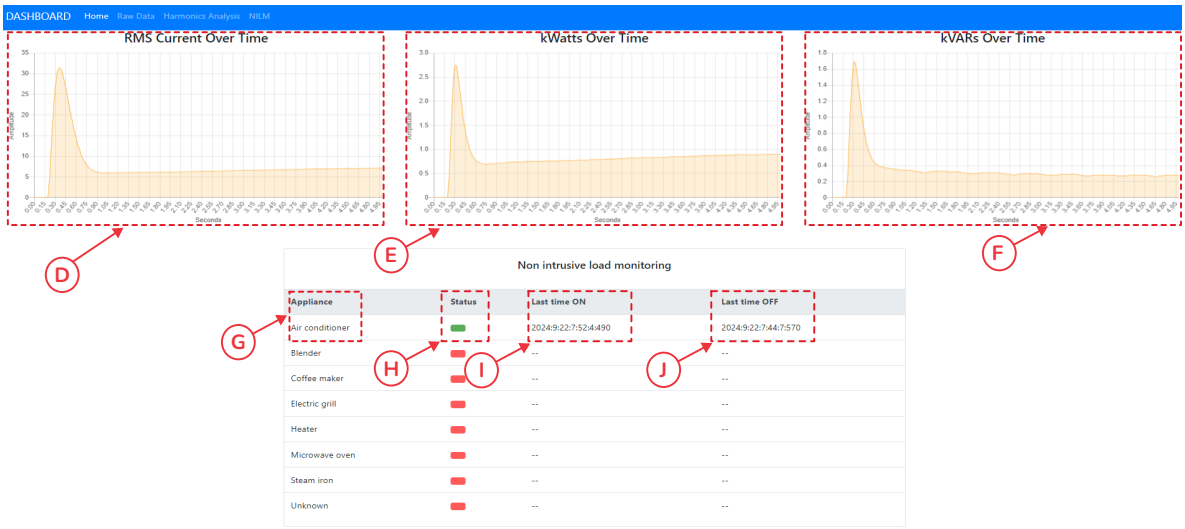


Figure 4.10: NILM tab displaying appliance-level signal waveforms and monitoring table: View D (RMS current chart), View E (active power chart), View F (reactive power chart), View G (recognized appliances), View H (ON/OFF status), View I (last activation time), and View J (last deactivation time).

Chapter 5

Experiments and Results

This chapter validates the proposed NILM system through real-world experiments, evaluating its event detection algorithm, unsupervised dataset generation, and deep learning classification models. The methodology targets four key aspects: First, data were collected from two households where seven appliances were manually switched on and off around 100 times each, allowing the smart meter to detect events in real time and store corresponding waveform segments. Second, an autoencoder compressed these signals, and a silhouette-optimized K-means algorithm clustered them to create unlabeled datasets. This process was tested progressively with 2 to 7 appliances to examine scalability. Third, three supervised models—1D CNN, WaveNet, and LSTM—were trained and evaluated on the generated datasets and the public NILM PLAID dataset using standard classification metrics. Finally, system efficiency was assessed via inference time, training data requirements, and cross-household experiments to evaluate generalization and practical deployment potential.

5.1 Experimental Setup Data Collection

To validate the proposed NILM system under realistic conditions, the custom-designed smart meter was installed at the main electrical panel of a residential household equipped with a split-phase power supply. The experimental setup involved seven commonly used household appliances: a mini-split air conditioner, a microwave oven, a coffee maker, a blender, an electric skillet, an electric heater, and an iron. These appliances were distributed across various rooms in the home to reflect a typical domestic setting. The technical information of appliances is presented in Table 5.1.

Each appliance was manually operated in its respective room location, which can

Table 5.1: Technical information of the appliances used in the experiments.

Appliance	Watts	Room	ON events	OFF events
Prime Mini-split A/C	1260	Bedroom	30	30
Daewoo Microwave oven	1500	Kitchen	100	100
Silex Coffee maker	900	Kitchen	100	100
Black and Decker Blender	550	Kitchen	100	100
Black and Decker Electric skillet	1250	Kitchen	100	100
Atvio Electric heater	800	Living room	100	100
Panasonic Iron	1500	Laundry	100	100

be seen in Table 5.1, by switching it ON and OFF approximately 100 times (except for the mini-split A/C, which was limited to 30 operations to avoid potential breakdown). During the experiments, each appliance remained ON for 10 seconds and OFF for 10 seconds, allowing the event detection algorithm to capture at least 5 seconds of the transient behavior associated with every switching event.

Throughout this process, the SM continuously measured a wide range of electrical parameters, including active and reactive power, RMS values, and power quality indicators. The real-time event detection algorithm embedded in the SM successfully identified the ON/OFF transitions, storing the corresponding waveform segments for further analysis. Figure 5.1 shows the installed smart meter and the selected appliances used in the experiment.

It is important to emphasize that the experimental setup was replicated in a second, completely different household with a distinct set of appliances. This was intentionally done to demonstrate the generality and scalability of the proposed NILM system beyond a single environment. To ensure consistency, the experiments in both households were performed following exactly the same procedures. For clarity of presentation and to avoid unnecessary repetition, the detailed description of experiments and results in the following sections correspond only to House 1, while the results from House 2—obtained under the same experimental protocol—are presented separately in Section 5.9. In this way, the experiments and outcomes for both households are clearly distinguished, allowing the reader to evaluate the system's performance across different conditions.

5.2 Electrical Parameters Selection

While the smart meter was configured to record the transients of RMS current, active power, and reactive power during event detection, an additional analysis was conducted to validate this selection. Principal Component Analysis (PCA) was applied to the

Figure 5.1: Testing environment: Smart meter installation and appliances used.

entire set of recorded electrical parameters to identify those contributing the most variance during the experiments. This aimed to determine which features offer the most informative representations of appliance behavior.

Figure 5.2 presents the results of the PCA, highlighting how certain features—such as total and fundamental RMS voltages and the Total Harmonic Distortion of current (THDi)—exhibit strong negative correlations in the first principal component. This suggests that while these parameters show high variance, they tend to decrease when other features increase. However, their usefulness for classification tasks is limited. For instance, the voltage drop observed when an appliance is activated is not intrinsic to the appliance itself but is instead a byproduct of the home's electrical wiring. Likewise, variations in THDi or fundamental RMS current are influenced by the presence of other simultaneous loads, reducing their reliability as standalone indicators.

In contrast, RMS current, active power, and apparent power—along with the power factor—demonstrate stronger discriminative capabilities. Among them, active and reactive power provide a concise yet effective summary of the energy behavior of appliances. This analysis confirms that the transients of RMS current, active power, and reactive power are sufficient to represent appliance signatures for NILM classification purposes.

5.3 Detected events

A total of 1260 switching events were detected during the data acquisition stage, equally divided between ON and OFF transitions. Figure 5.3 presents the normalized active

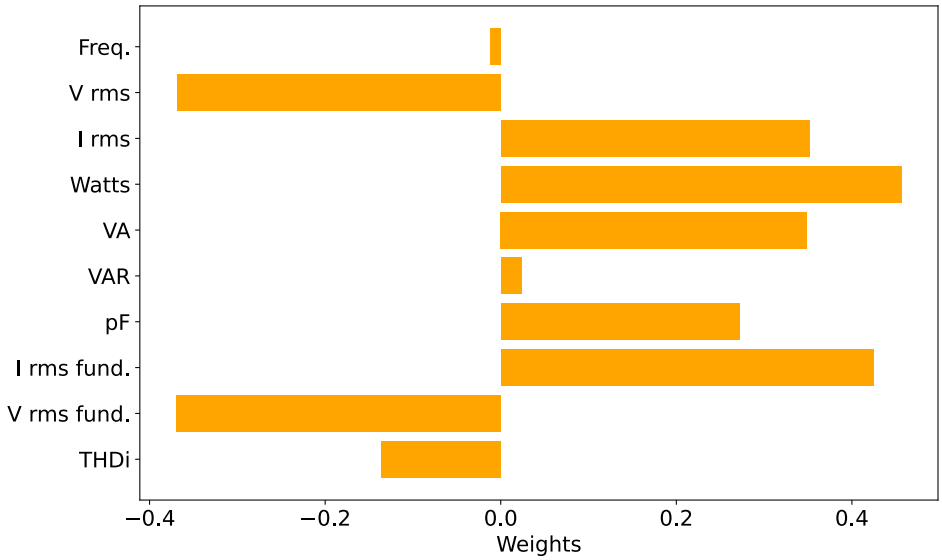


Figure 5.2: Weights of the electrical parameters in the first principal component.

power trajectories associated with each event across the different appliances, plotted over a 5-second window. Each subplot shows the sequence of switching instances for one appliance, with red arrows indicating the typical ON and OFF points.

The consistent overlap of trajectories for each appliance suggests that the system captures stable and repeatable power signatures. Loads such as the heater and steam iron exhibit highly uniform patterns, indicating minimal intra-class variability. On the other hand, more dynamic transients can be observed in appliances like the microwave and air conditioner, likely due to multi-phase startup behavior or internal control cycles.

These results confirm that the smart meter reliably detects characteristic transients during switching events and that these patterns offer sufficient discriminative information to support unsupervised clustering and classification tasks. To enable the training and evaluation of future algorithms, the detected events of Figure 5.3 were organized into two separate unlabeled datasets: one corresponding to appliance ON events and the other to OFF events. Each dataset exhibits a three-dimensional structure with the shape $(630, 500, 3)$, where the first dimension represents the number of samples, the second corresponds to the time points captured for each event, and the third dimension contains the electrical features (RMS current and active and reactive power). A representation of the dataset structure is provided in Figure 5.4, which illustrates the arrangement of samples, time windows, and features. It should be noted that this arrangement may vary depending on the input shape required by the specific AI model.

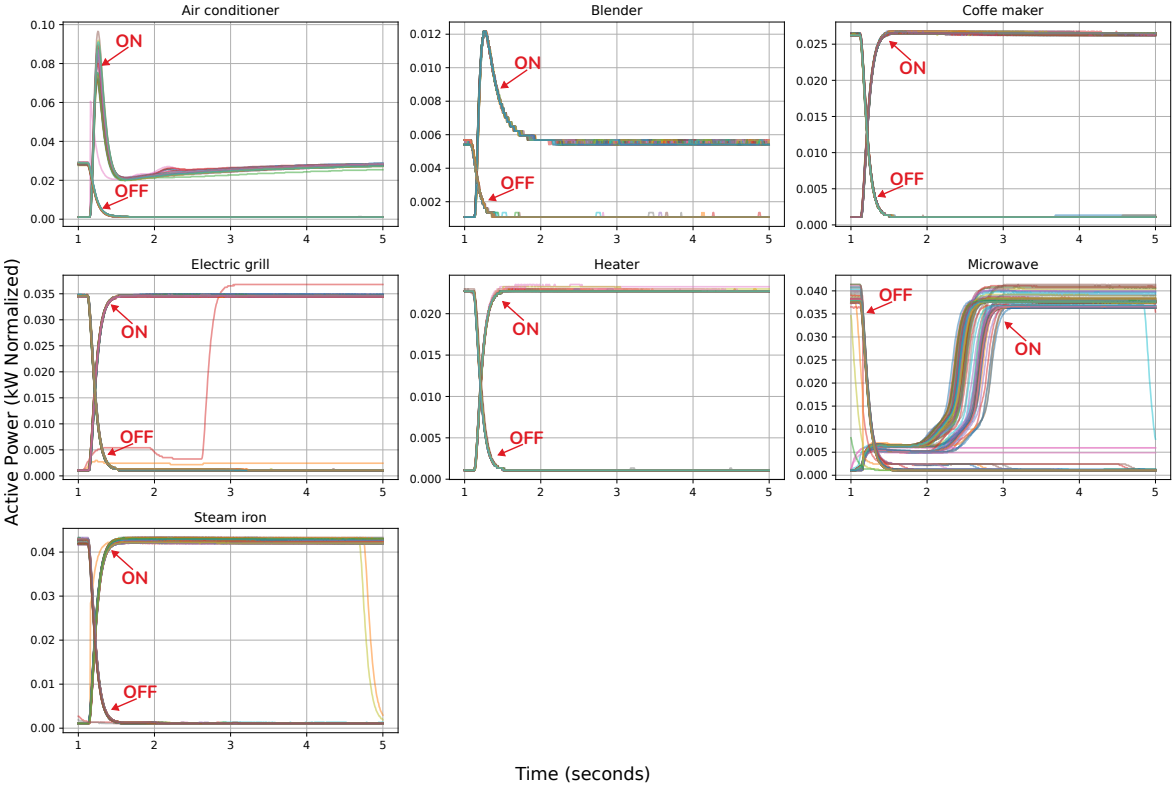


Figure 5.3: Normalized active power trajectories during ON and OFF events.

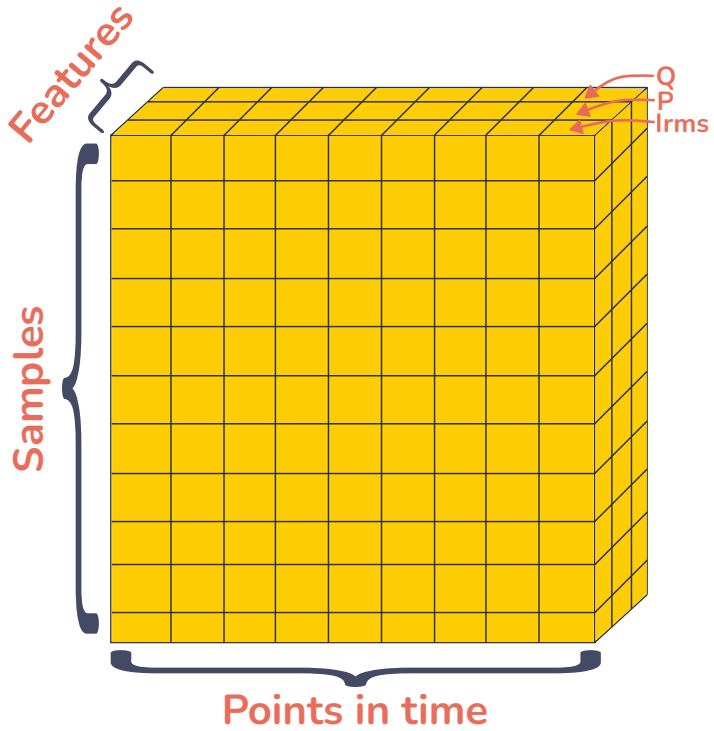


Figure 5.4: Representation of the dataset structure for ON and OFF events.

5.4 Feature Extraction via Autoencoder

The 1D-CNN Autoencoder architecture, illustrated in Figure 4.4, was employed to extract latent features from time series signals of detected ON/OFF events by learning abstract representations directly from raw data. Unlike approaches based on handcrafted features, this model captures the essential temporal dynamics through a fully data-driven process. The encoder compresses the input by identifying common patterns across events, while the decoder reconstructs the original signal from the latent representation. The accuracy of this reconstruction indicates that the encoded features effectively retain the most relevant information, serving as both a validation of feature quality and an indirect measure of the model's ability to generalize across unseen data.

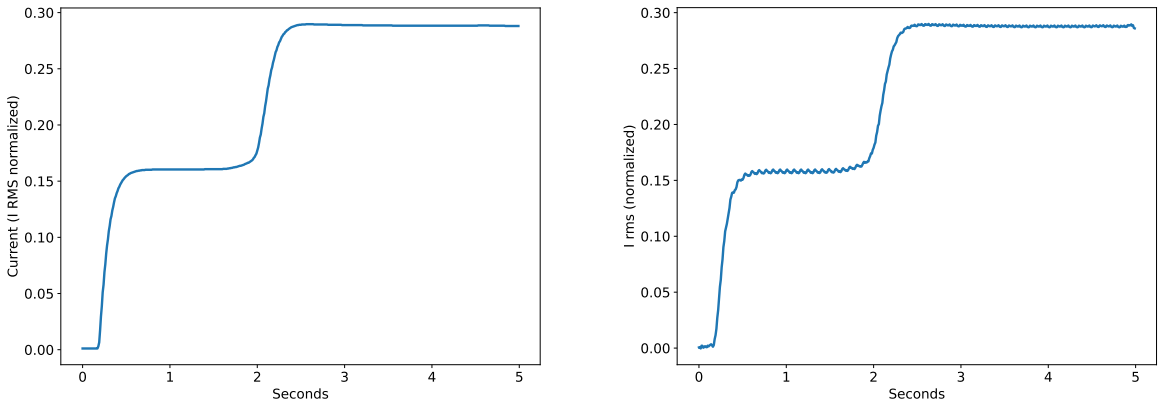
As illustrated in Figure 5.5, the reconstructed signals closely match the original inputs, confirming that the learned latent space retains the core structure and behavior of the electrical events. This visual evidence is supported by a quantitative evaluation using an R-squared metric, where the model achieved a score of 0.99 on the test set, based on an 80/20 training and test partition. The model was trained over 20 epochs, which proved sufficient to reach convergence without any signs of overfitting. This high level of accuracy suggests that the encoding preserves sufficient detail for downstream tasks while maintaining generalization.

By translating each event into a compact feature vector, the Autoencoder facilitates efficient clustering and classification in later stages of the NILM system. These latent vectors are especially valuable for distinguishing subtle differences between appliance signatures. The resulting feature space thus provides a robust foundation for unsupervised learning, enabling the system to identify appliance usage without labeled data.

5.5 Unsupervised Clustering and Dataset Creation

Once the latent representations were extracted from the detected events through the Autoencoder model, an unsupervised clustering process was performed to group similar events without requiring prior labels. The K-means clustering algorithm was applied to the latent space, with the number of clusters automatically determined using the Silhouette method, as previously described.

For the ON events, the optimal number of clusters identified was seven, which matches the number of appliances monitored in the experiment. Figure 5.6a presents the latent space representations projected into two dimensions using PCA for visualization.



(a) Original ON event. (b) Reconstructed ON event.

Figure 5.5: Comparison between the original event and reconstructed event from latent space using 1-D CNN Autoencoders.

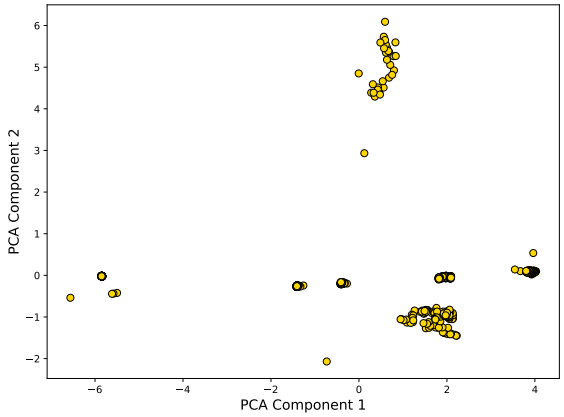
The resulting clusters are depicted in Figure 5.6b, where clear separation among the seven clusters can be observed. Given that the ground truth appliance labels were known, the clustering performance could be quantitatively evaluated, yielding an accuracy of 99.23%, a recall of 99.31%, a precision of 99.31%, and a F1-score of 99.30%.

For the OFF events, the same number of centroids identified from the ON events was used to maintain consistency. The PCA projection of the latent representations for the OFF events is shown in Figure 5.6c, while the clustering results are illustrated in Figure 5.6d. Despite the potentially lower distinctiveness of OFF events, the clustering still achieved high performance, with an accuracy of 98.11%, a recall of 98.21%, a precision of 98.46%, and a F1-score of 98.24%.

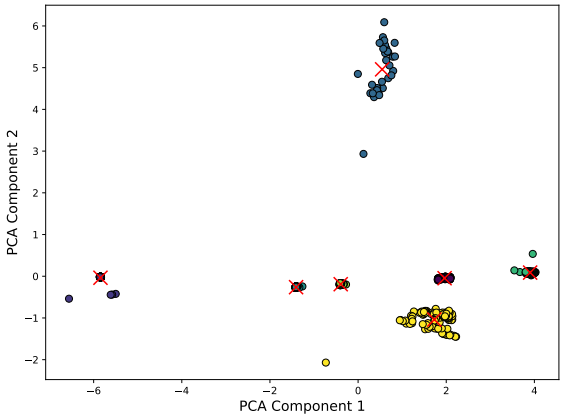
The clustered ON and OFF events were subsequently integrated into a single labeled dataset. Each cluster was assigned a numerical label ranging from 0 to 13, reflecting the total of fourteen clusters identified: seven corresponding to ON events and seven to OFF events. This unified and labeled dataset serves as the input for the supervised learning models described in Section 4.5.

To further evaluate the effectiveness of the clustering process and the automatic selection of the number of clusters through the silhouette method, two additional experiments were conducted. In both cases, the number of clusters was determined solely using the ON events, as these are generally more distinctive and easier to separate.

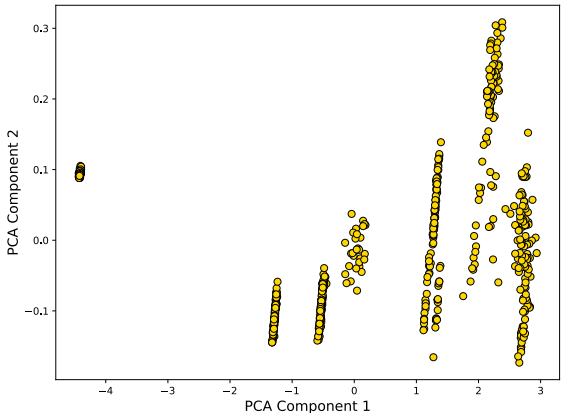
In the first experiment, the clustering process was repeated progressively, starting with only two appliances and then gradually increasing the number to three, four, and so on, until all seven appliances were included. For each case, clustering was performed using the latent space representations obtained during feature extraction,



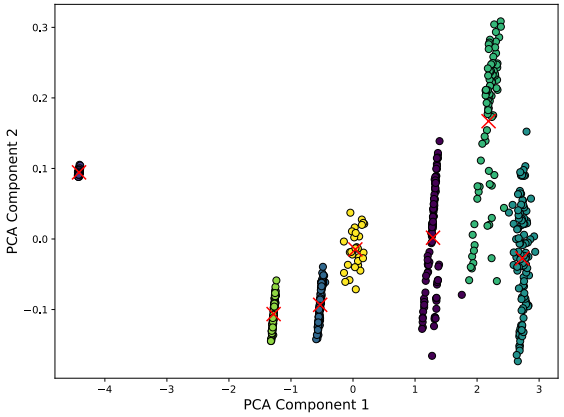
(a) PCA projection of latent representations (ON events).



(b) K-means clustering results (ON events).



(c) PCA projection of latent representations (OFF events).



(d) K-means clustering results (OFF events).

Figure 5.6: PCA visualization of latent representations from ON and OFF events of the seven appliances and the corresponding K-means clustering results.

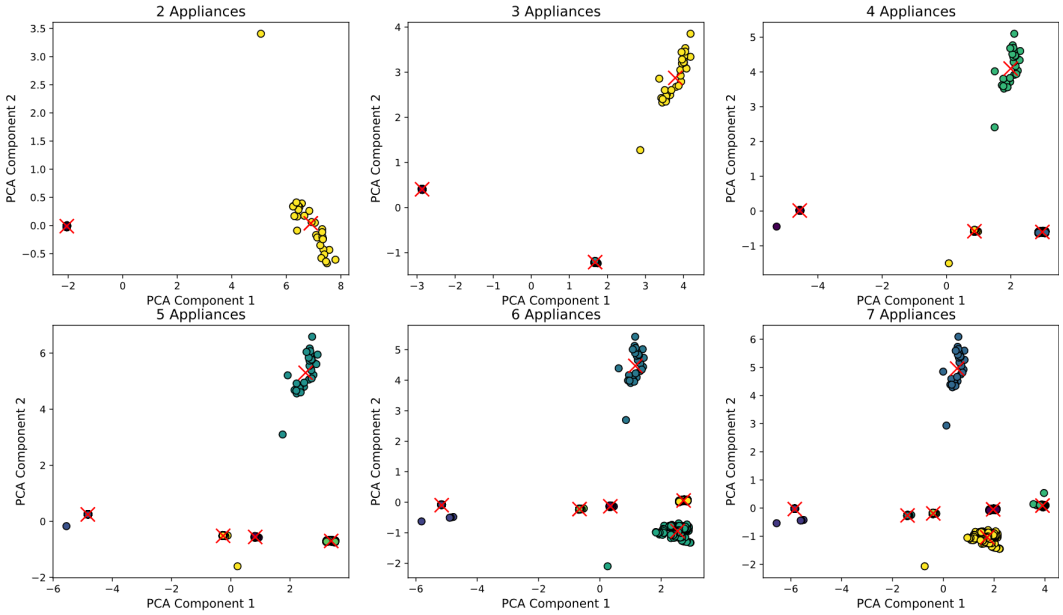


Figure 5.7: Effect of increasing the number of appliances on the automatic selection of the number of clusters.

and the silhouette method was used to automatically select the optimal number of clusters. As shown in Figure 5.7, the clustering results accurately reflect the growing number of appliances, with the correct number of clusters identified in each scenario. Additionally, the clustering accuracy remained consistently high throughout the experiment, achieving 100% accuracy with two and three appliances, followed by 99.41% with four, 99.54% with five, 99.08% with six, and 99.22% with seven appliances. The second experiment assessed the robustness of the automatic K selection by analyzing all possible unique combinations of the seven appliances, resulting in a total of 120 different tests. For each combination, the number of clusters was automatically selected using the silhouette score. Figure 5.8 compares the actual number of appliances present in each combination with the number of clusters identified by the algorithm. The method successfully matched the correct number of clusters in 88% of the cases, demonstrating its strong ability to generalize and adapt to varying appliance combinations.

Together, these experiments confirm the reliability of the clustering process and its ability to automatically adapt to different scenarios, without requiring manual tuning of the number of clusters.

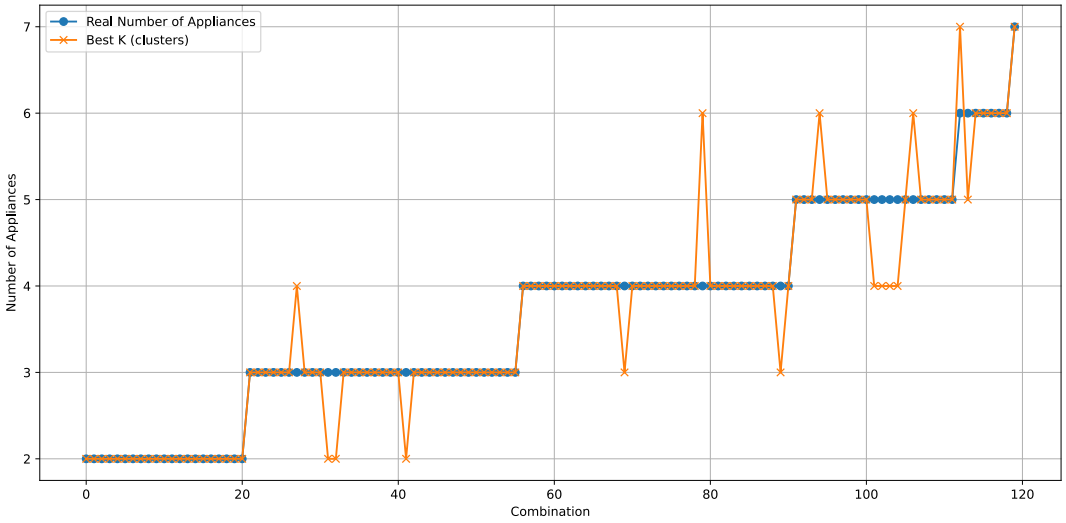


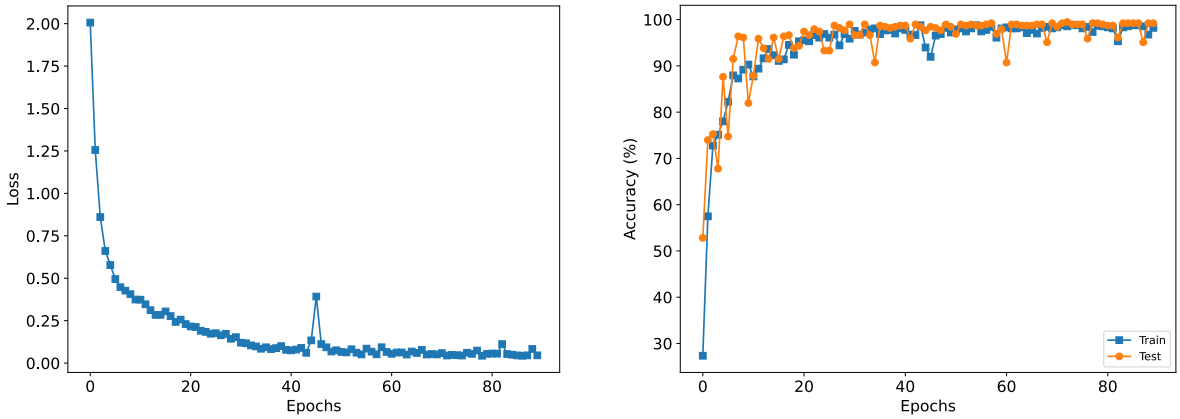
Figure 5.8: Performance of how automatic K cluster selection reflects the actual number of appliance combinations.

5.6 Classification with Deep Learning Models

Using the previously generated dataset with the automatic clustering technique, the models presented in Section 4.5 were tested to classify the ON and OFF events from the seven appliances. The results for the 1D-CNN model are discussed first, while the performance of the remaining models will be addressed later in this subsection.

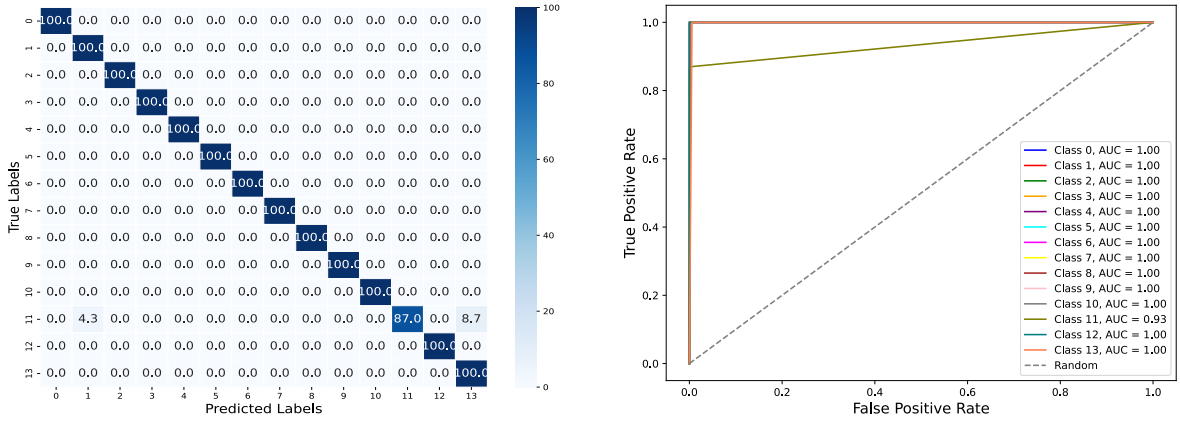
The 1D-CNN model, whose architecture is displayed in Figure 4.5, was trained using an 80/20 training and test split. The training process lasted 90 epochs, and the evolution of the loss and accuracy throughout the training is presented in Figure 5.9. Specifically, Figure 5.9a shows the progressive reduction of the loss function over the epochs, with rapid convergence during the initial phase and stabilization as training progresses. Similarly, Figure 5.9b illustrates the accuracy trends, where both the training and test datasets achieved stable performance above 95% after approximately 20 epochs, with minor fluctuations observed. Once the model was trained, its classification effectiveness was evaluated. Figure 5.10a presents the confusion matrix, highlighting a high rate of correct classifications across nearly all appliance classes, with only minimal misclassifications. Additionally, Figure 5.10b displays the Receiver Operating Characteristic (ROC) curves for each class, showing that most appliance classes achieved an Area Under the Curve (AUC) close to or equal to 1.0, reflecting excellent classification performance.

The final evaluation metrics of the 1D-CNN model indicate a classification accuracy of 99.23%, a precision of 98.72%, a recall of 99.06%, and an F1-score of 98.82%. These results are summarized in Table 5.2, confirming the model's capability to reliably



(a) Loss over epochs (b) Accuracy over epochs

Figure 5.9: Training behavior of the 1D-CNN model in terms of loss and accuracy.



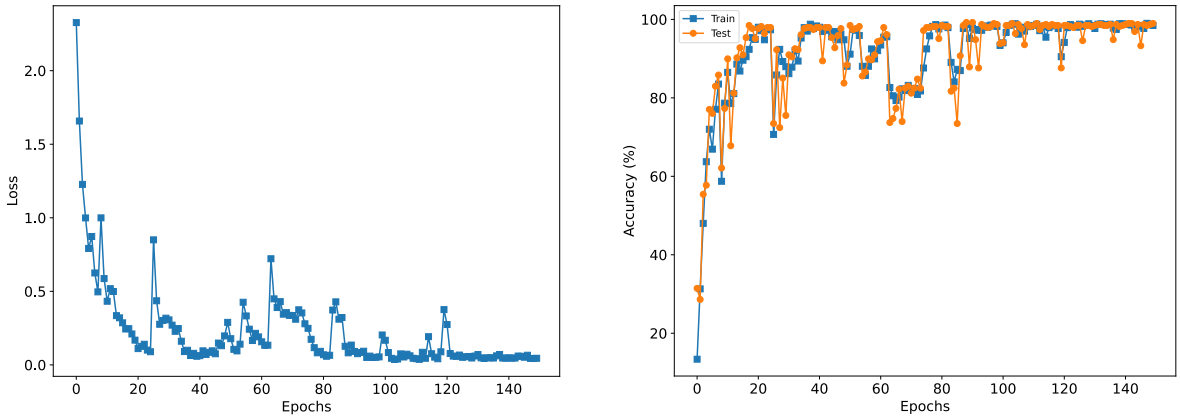
(a) Percentage Confusion Matrix (b) ROC curve

Figure 5.10: Classification performance of the 1D-CNN model.

identify appliance events using the features extracted during the previous stages. The next model evaluated is based on the LSTM architecture, as illustrated in Figure 4.7. The training and testing were conducted using the same dataset obtained in the clustering section, following an 80/20 split for training and testing, respectively.

Figure 5.11 shows the training behavior of the LSTM model. In Figure 5.11a, the loss over epochs is presented, where a noticeable reduction is observed as training progresses, though with certain oscillations due to the nature of recurrent architectures. In parallel, Figure 5.11b illustrates the accuracy over epochs for both the training and testing datasets. The model consistently achieved high accuracy throughout the training process, with the test accuracy gradually stabilizing near its highest values.

The classification performance of the LSTM model is detailed in Figure 5.12. The confusion matrix, displayed in Figure 5.12a, indicates a strong classification performance across most classes, with only minor misclassifications in some appliance



(a) Loss over epochs (b) Accuracy over epochs

Figure 5.11: Training behavior of the LSTM model in terms of loss and accuracy.

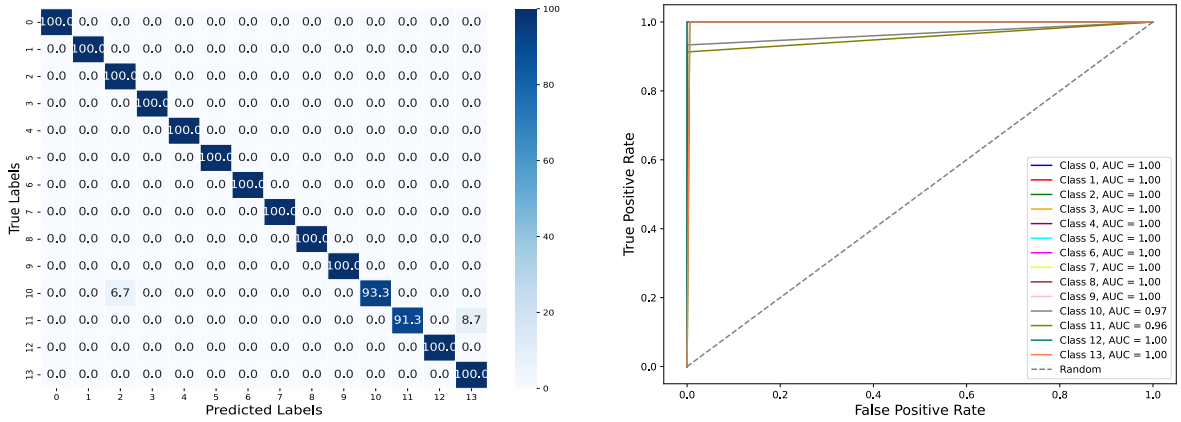
categories. Additionally, the ROC curves for each class are shown in Figure 5.12b, where most classes achieved an AUC of 1.00, except for classes 10 and 11 with slightly lower values, which still demonstrate excellent separability.

Quantitatively, the LSTM model achieved an accuracy of 98.97%, a precision of 99.21%, a recall of 98.90%, and an F1-score of 99.02%, as reported in Table 5.2.

The performance of the WaveNet-based model, depicted in Figure 4.6, was also assessed using the same dataset partition described earlier. The training dynamics of this model are presented in Figure 5.13. The loss curve in Figure 5.13a highlights a consistent decrease in the training loss, though periodic fluctuations were observed throughout the process. Despite these fluctuations, the loss steadily approaches low values. The accuracy plot in Figure 5.13b demonstrates that both training and testing accuracies remained consistently high after the initial epochs, ultimately converging to stable values near 100%.

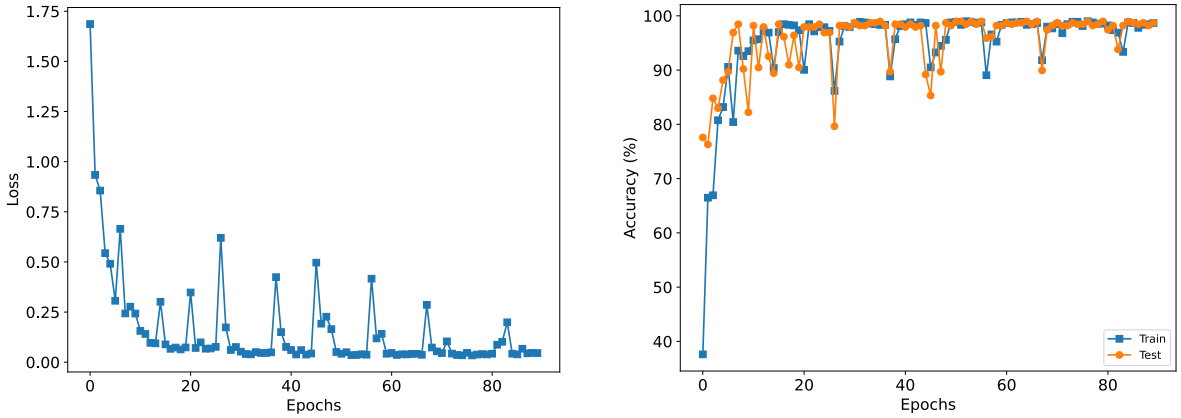
The classification results are summarized in Figure 5.14. The confusion matrix in Figure 5.14a shows that most appliance classes were correctly identified, with only a few instances of misclassification concentrated in class 11. The ROC curves in Figure 5.14b confirm the strong classification capability of the model, as most classes achieved an AUC close to or equal to 1.00.

In terms of overall metrics, the WaveNet model attained an accuracy of 98.71%, a precision of 99.10%, a recall of 98.44%, and an F1-score of 98.65%, as summarized in Table 5.2. To further evaluate the generalization performance of the tested models, a 5-fold cross-validation experiment was conducted. The results, presented in Figure 5.15, compare the test accuracy of the LSTM, 1D-CNN, and WaveNet models across all five folds. Despite some variation in individual fold performance, all models



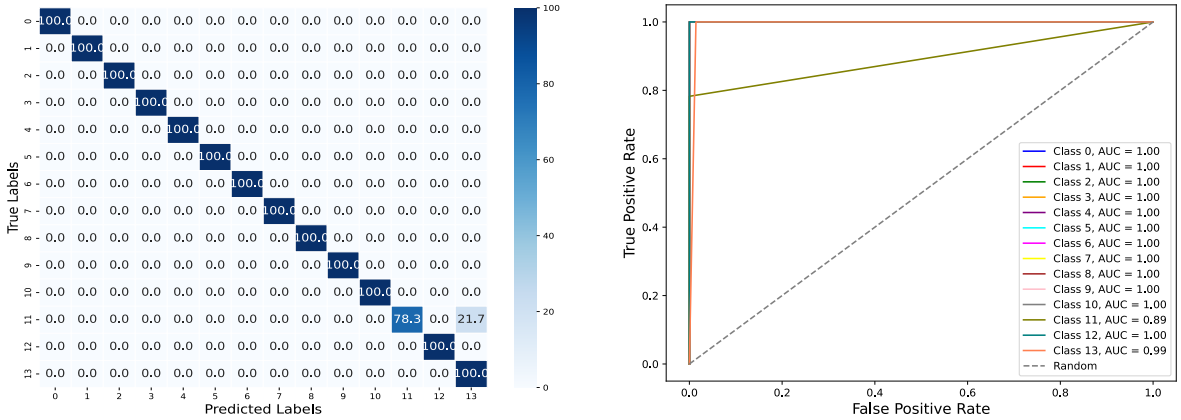
(a) Percentage Confusion Matrix (b) ROC curve

Figure 5.12: Classification performance of the LSTM model.



(a) Loss over epochs (b) Accuracy over epochs

Figure 5.13: Training behavior of the WaveNet model in terms of loss and accuracy.



(a) Percentage Confusion Matrix (b) ROC curve

Figure 5.14: Classification performance of the WaveNet model.

Table 5.2: Classification performance of the tested models.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
1D-CNN	**99.23**	98.72	**99.06**	98.82
LSTM	98.97	**99.21**	98.90	**99.02**
WaveNet	98.71	99.10	98.44	98.65

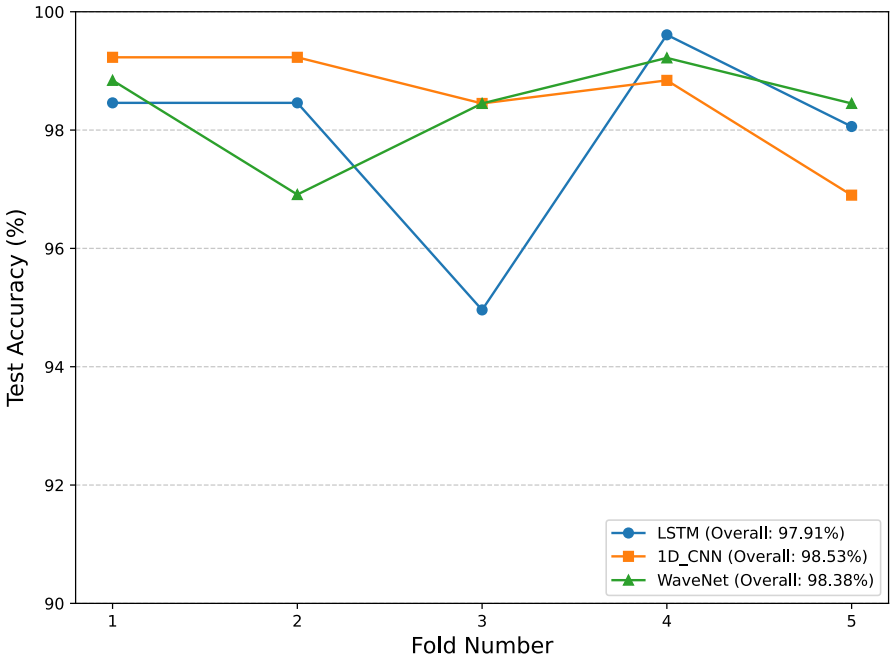


Figure 5.15: Comparison of LSTM, 1D-CNN, and WaveNet Models in a 5-Fold Cross-Validation Test.

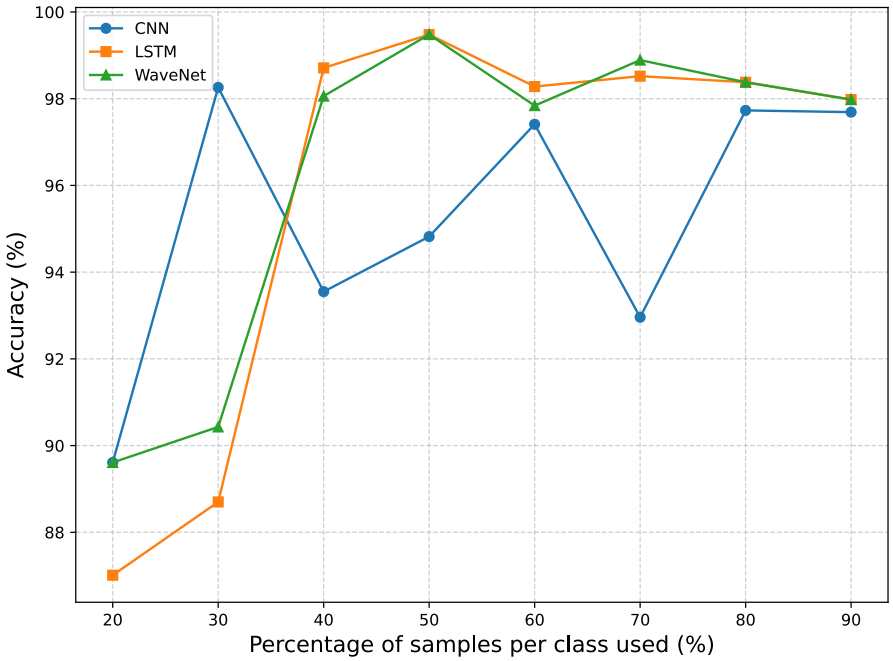


Figure 5.16: Effect of varying the percentage of samples per class on the classification accuracy of the 1D-CNN, LSTM, and WaveNet models.

consistently achieved high accuracy levels. The 1D-CNN model demonstrated the most stable performance, with an average accuracy of 98.53%. The WaveNet model followed closely with 98.38%, while the LSTM model achieved 97.91%. These results confirm the robustness of all three models and indicate that the proposed classification framework generalizes well across different data partitions.

Also, an experiment was conducted to evaluate how the number of training samples per class affects model performance. In this test, different subsets of the full dataset were created by selecting increasing percentages of samples per class, ranging from 20% to 90%. For each subset, the standard 70/30 training and testing split was maintained. As shown in Figure 5.16, all models demonstrated noticeable improvements in accuracy as more data became available. At lower data availability (20–30%), the 1D-CNN performed more robustly than the LSTM and WaveNet models. However, starting at around 40%, both LSTM and WaveNet models began to outperform the CNN, with their accuracy converging near 99% when 60% or more of the total samples were used. This indicates that while 1D-CNNs may be better suited for scenarios with limited data, LSTM and WaveNet architectures benefit significantly from larger datasets and exhibit superior generalization once a sufficient number of samples per class is available.

5.7 Computational Performance

To provide a comprehensive comparison among the tested deep learning models, computational efficiency was also evaluated. All experiments were conducted on a machine equipped with an AMD Ryzen 9 6900HS processor and an NVIDIA GeForce RTX 3060 Laptop GPU. Table 5.3 summarizes the computational characteristics of the 1D-CNN, LSTM, and WaveNet models. These insights, combined with the accuracy results presented earlier, allow for a well-rounded analysis of each approach's strengths and limitations.

The 1D-CNN model demonstrated excellent classification performance, achieving the highest accuracy (99.23%) and strong generalization, as confirmed by its average cross-validation score of 98.53%. Additionally, it was the most efficient in terms of training and inference times, completing training in just 27.14 seconds and performing inference in 0.0026 milliseconds per sample. These features make it highly suitable for real-time applications. However, its simplicity comes with a cost in model size: the 1D-CNN had the largest number of trainable parameters (over 2.2 million), resulting in a memory footprint of 12.82 MB. The LSTM model, while slightly less accurate (98.97%), offered the best balance between performance and efficiency. It achieved the highest F1-score (99.02%) and had the smallest number of parameters (73,274), leading to a compact memory size of just 8.97 MB. In terms of computational requirements, it maintained a low inference time comparable to the 1D-CNN (0.0026 ms/sample), although its training time was longer than the 1D-CNN model (41.89 seconds). These characteristics make the LSTM particularly attractive for deployment in embedded systems or low-power environments, especially when model size is a critical constraint.

In contrast, the WaveNet model—while competitive in accuracy (98.71%) and

Table 5.3: Computational performance comparison of the tested models.

Metric	1D-CNN	LSTM	WaveNet
Training time (s)	**27.14**	41.89	127.61
Model parameters	2,226,584	**73,274**	1,194,389
Inference time (batch, s)	**0.001004**	0.001014	0.005034
Inference time per sample (ms)	**0.0026**	**0.0026**	0.0194
Trainable params	2,226,584	**73,274**	1,194,389
Non-trainable params	0	0	0
Total mult-adds	**46.04 MB**	271.51 MB	2.36 GB
Input size (MB)	0.10	0.10	0.10
Fwd/back pass size (MB)	**3.81**	8.58	96.98
Params size (MB)	8.91	**0.29**	4.78
Estimated total size (MB)	12.82	**8.97**	101.85

cross-validation performance (98.38%)—was significantly more resource-intensive. Its architecture, which includes dilated causal convolutions, provides a high capacity for learning complex temporal dependencies, which is reflected in its near-perfect ROC curves. However, this complexity results in substantial computational overhead: training took 127.61 seconds, and inference was notably slower (0.0194 ms/sample). Furthermore, the model performed over 2.36 billion multiply–add (mult-add) operations, a metric that quantifies the total number of multiplication and addition steps required during computation. This high count, along with a memory footprint of 101.85 MB, underscores the model's demand for processing power and makes it less practical for deployment in real-time or resource-limited scenarios.

5.8 Dataset Adaptability in Supervised Learning Model

As discussed in Section 5.5, the self-adaptive capability of the k-means algorithm with automatic centroid selection was evaluated by varying the number of appliances and testing all possible appliance combinations. This experiment demonstrated how effectively the method could infer the number of events present in an unlabeled dataset under different configurations.

To extend this adaptability analysis to a supervised DL model, all 120 possible combinations of appliance presence (considering both ON and OFF events) were used to retrain and evaluate an LSTM-based model. This particular architecture was chosen given its balance between computational efficiency and classification accuracy, as previously evidenced in Table 5.3.

The results of this experiment are presented in Figure 5.17. After applying a 70/30 train–test split across all combinations, the model depicted in Figure 5.12 achieved overall performance scores of 98.80% accuracy, 98.87% precision, 98.90% recall, and 98.73% F1-score. Notably, only four appliance combinations resulted in an accuracy below 90%. These findings highlight the robustness and adaptability of the LSTM-based model when faced with diverse appliance configurations.

5.9 Cross-Household Generalization

To assess the robustness and generalization ability of the proposed NILM system, all experiments were replicated in a second residential environment. The designed smart

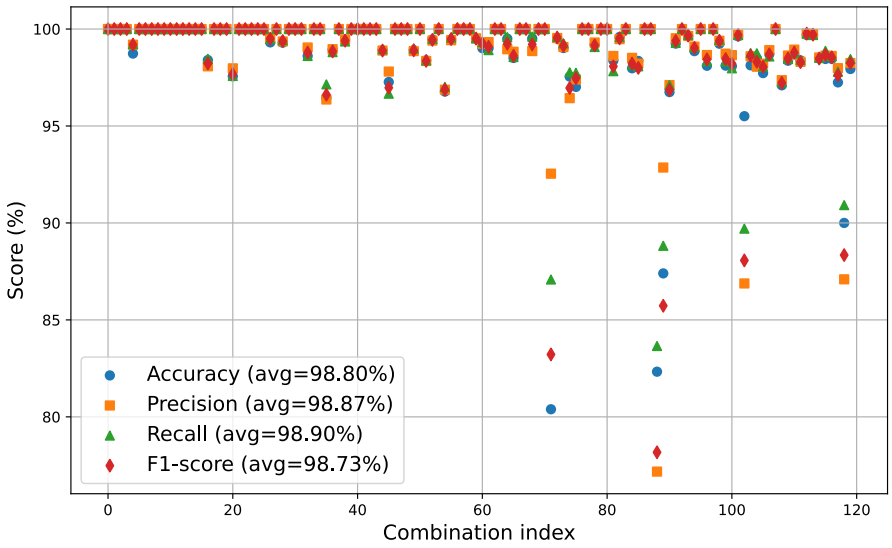


Figure 5.17: Classification performance of the LSTM model across the 120 appliance combinations.

meter was installed in House 2, which, as in House 1, uses a split-phase electrical configuration. In House 2, seven distinct appliances—*blender*, *grinder*, *induction oven*, *microwave oven*, *toaster*, and *vacuum*—were systematically switched ON and OFF 100 times each. During each cycle, appliances remained ON for 10 seconds and OFF for 10 seconds, ensuring that the event detection algorithm captured at least 5 seconds of transient behavior in the RMS current as well as in the active and reactive power for every switching event. This procedure generated 200 events per appliance, as illustrated in Figure 5.18.

All experiments were repeated from scratch: the event detection algorithm was applied to extract ON and OFF switching events, followed by clustering with automatic k selection to group similar signatures to create a tailored dataset, and finally appliance classification using the LSTM model introduced in Figure 4.7, which had shown a favorable balance between accuracy and computational efficiency in prior evaluations.

The clustering algorithm continued to perform exceptionally well in this new environment. For OFF events, the clustering yielded perfect scores across all metrics, achieving 100.00% accuracy, recall, precision, and F1-score. For ON events, the performance remained strong, with an accuracy of 99.71%, recall of 99.72%, precision of 99.74%, and F1-score of 99.73%. These results confirm that this clustering approach is highly stable and reliable across different household conditions, which is further illustrated by the clear cluster separation observed in the PCA projections of Fig. 5.19.

Moreover, the automatic estimation of the number of clusters (k) was re-assessed

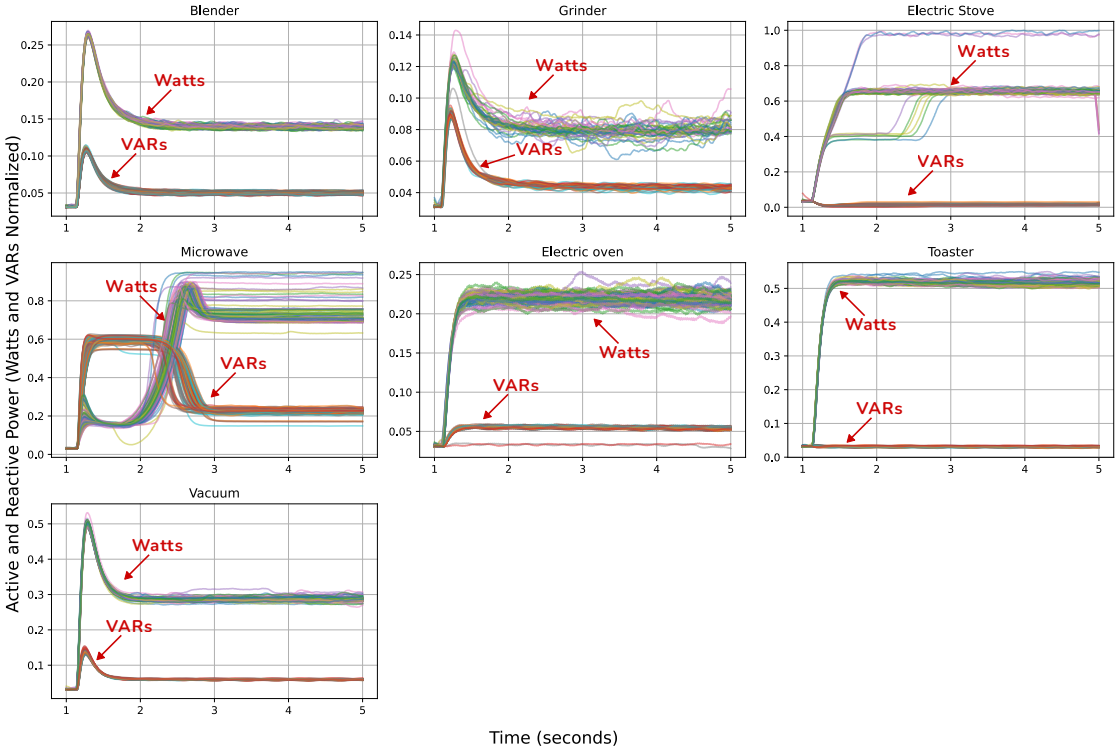
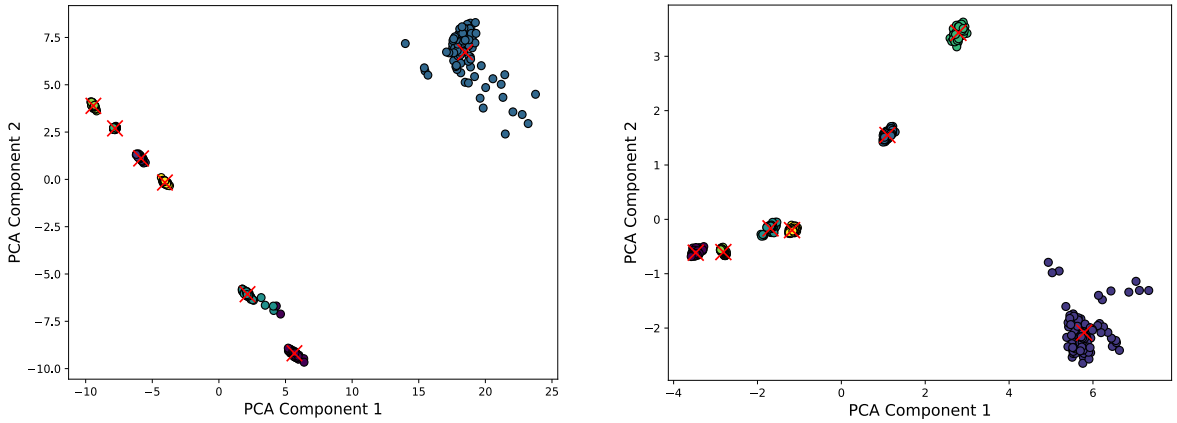


Figure 5.18: ON events generated in House 2 for the seven appliances, corresponding to their active and reactive power signatures.



(a) Clustering outcomes of ON events across seven appliance classes.

(b) Clustering outcomes of OFF events across seven appliance classes.

Figure 5.19: Clustering of ON and OFF events using k-means with automatic cluster selection, visualized through a two-dimensional PCA projection.

Table 5.4: Clustering and classification results for House 2, including LSTM performance with cross-validation and k-selection accuracy.

Experiment	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Clustering OFF Events	100.00	100.00	100.00	100.00
Clustering ON Events	99.47	99.48	99.43	99.44
LSTM Classification	99.31	99.38	99.19	99.26
k **Selection Accuracy (120 combinations):** 90.83%				

over all 120 possible combinations of appliances, with the correct number of clusters being identified in 90.83% of the cases. This outcome underscores the adaptability and robustness of the unsupervised k-selection mechanism when exposed to new environments and varied appliance mixes.

For the final classification task, the LSTM model was evaluated on the complete dataset, which combined the seven ON-event classes and the seven OFF-event classes, resulting in a total of 14 classes. Using a cross-validation strategy, with five folds the LSTM model achieved an overall accuracy of 99.47%, with a recall of 99.43%, precision of 99.48%, and F1-score of 99.44%. These results demonstrate the system's capacity to generalize effectively to unseen households, maintaining high performance in both unsupervised and supervised stages. A summary of the key results is presented in Table 5.4. To further validate the generalizability of the proposed NILM system, an external benchmark was conducted using the public PLAID dataset. Despite inherent challenges—such as variability in sampling setups and appliance location inconsistencies across houses—the PLAID dataset provides a valuable reference since it contains measurements from multiple locations and includes 17 different appliance classes. The dataset consists of submetered ON events recorded over 42 cycles at a fundamental frequency of 60 Hz, with raw current and voltage sampled at 30 kHz [115].

Since the LSTM model of Figure 4.7 was selected for deployment in the web application, it was also tested on PLAID events. Specifically, the subset corresponding to the CMU Lab location was chosen, as it provided the largest number of appliances and recorded events per appliance compared to the other locations, offering a richer and more balanced evaluation set. This subset includes 11 appliance classes: *compact fluorescent lamp*, *fridge*, *vacuum*, *coffee maker*, *incandescent light bulb*, *water kettle*, *fan*, *air conditioner*, *hair iron*, *laptop*, *soldering iron* and *blender*.

As the PLAID dataset provides raw current and voltage signals, pre-processing was required to align with the model's input representation. RMS current, active power, and reactive power were computed from the 30 kHz waveforms, resulting in transient ON-event signatures with a duration of approximately 0.7 seconds, as illustrated in

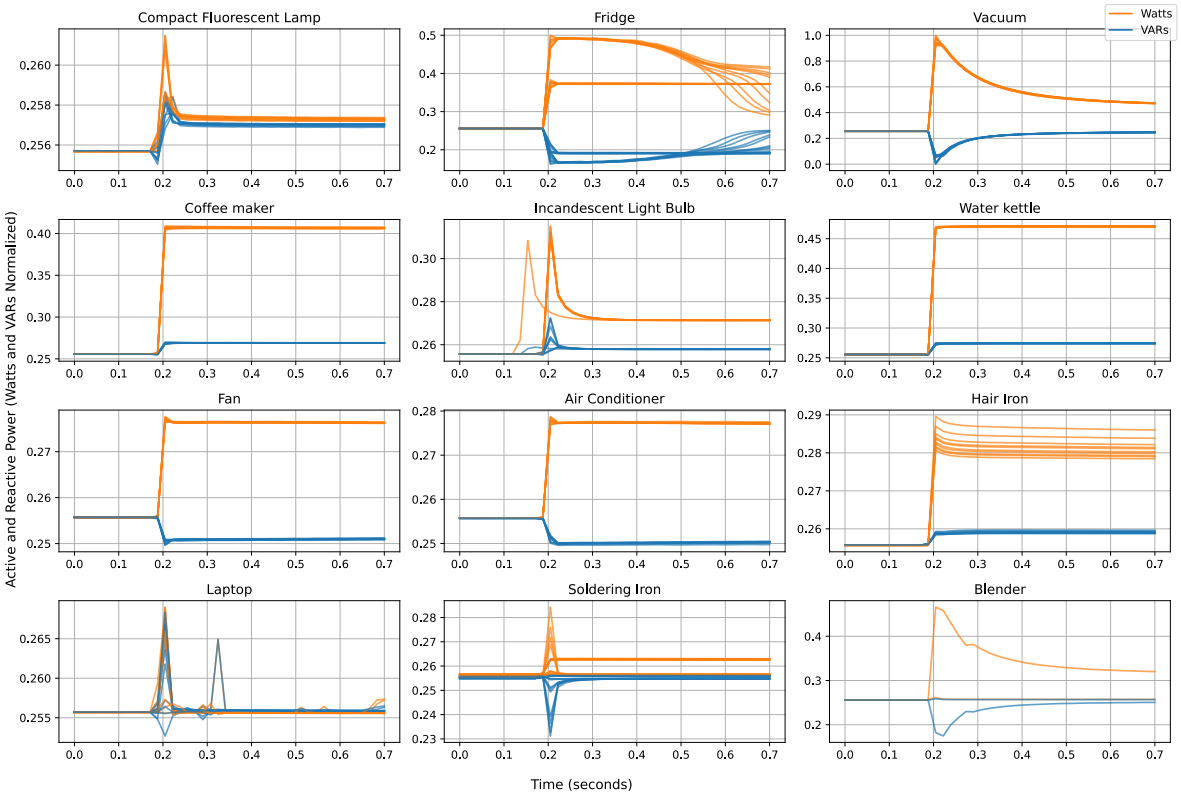


Figure 5.20: Transient ON-event signatures computed from PLAID data after preprocessing the raw voltage and current signals into RMS current, active power, and reactive power.

Figure 5.20. To ensure a fair evaluation, the blender appliance class was excluded due to its extremely limited availability (only two events). The model was trained and evaluated using a 70/30 train-test split, following the same methodology as in the previous experiments. The proposed LSTM achieved an accuracy of 96.15%, recall of 97.30%, precision of 97.16%, and F1-score of 96.86%, confirming that it maintains competitive performance even on external datasets with unseen appliances and different acquisition conditions. A per-class performance breakdown is provided in the confusion matrix of Figure 5.21, which highlights the classification capability across the diverse set of appliance events. Additionally, the PLAID dataset with the LSTM model was also tested with five-fold cross-validation, yielding mean values across the folds of 97.54% test accuracy, 97.86% precision, 97.54% recall, and 97.49% F1-score.

5.10 Event Classification Web Response Time

To evaluate the latency between the detection of an electrical event by the smart meter and its classification by the server, an experiment was conducted using the real-time

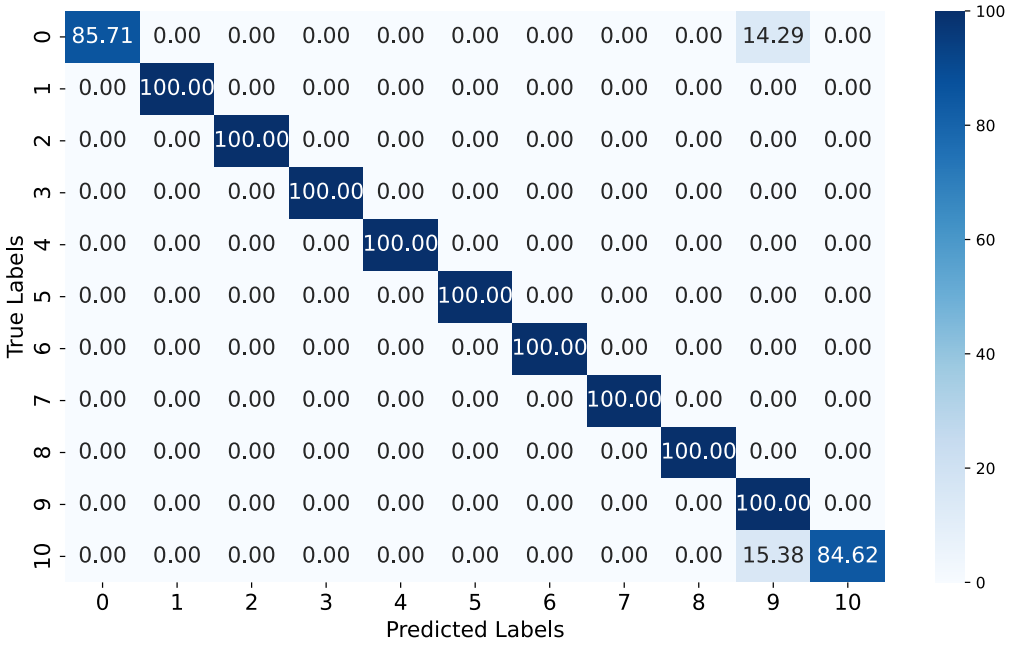


Figure 5.21: Confusion matrix of the LSTM model evaluated on the PLAID dataset (CMU Lab subset).

detection and classification infrastructure developed in this work. The smart meter was configured to record the UTC timestamp at the moment an ON event was detected. This timestamp was then transmitted through the IoT pipeline to the web server, where the event was processed and classified.

The web application was deployed using the Heroku cloud platform, as detailed in Chapter 3. Specifically, the application operated under the dyno *General Purpose 4CPU-16GB* configuration, which provides 16GB of RAM and 4x dedicated vCPUs. These specifications allowed for consistent background processing of incoming MQTT messages and near real-time classification of events through the DL learning model.

Table 5.5 presents 10 event samples from the experiment. Each row includes the timestamp when the event was detected by the smart meter, the timestamp when the event was classified on the server, and the calculated delay Δt between these two points. For the entire experiment, the calculated Mean Δt was **1.05 s**.

Table 5.5: Event detection and classification response times.

Sample	Detection Time (Smart Meter)	Classification Time (Server)	Δt (s)
1	03:54:14.800	03:54:15.876	1.08
2	03:54:24.870	03:54:25.935	1.07
3	03:54:34.180	03:54:35.260	1.08
4	03:54:44.970	03:54:46.002	1.03
5	03:54:55.830	03:54:56.862	1.03
6	03:55:07.400	03:55:08.438	1.04
7	03:55:31.760	03:55:32.799	1.04
8	03:55:41.390	03:55:42.405	1.01
9	03:56:06.640	03:56:07.718	1.08
10	03:56:19.920	03:56:21.013	1.09

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this work, the design and implementation of an IoT-based Non-Intrusive Load Monitoring system capable of real-time operation in real residential environments is introduced. The proposed system integrates event detection for ON and OFF states, unsupervised learning for automatic appliance dataset generation, and deep learning-based classification for load classification. Additionally, a web-based interface was developed to provide remote monitoring and event visualization, while the system's deployment in actual households demonstrated its adaptability to different electrical consumption profiles.

Building upon this, the integration of a real-time event detection and disaggregation algorithm within the smart meter enabled the precise identification of appliance switching events from aggregated load signals. During the experiments, all the ON and OFF events generated by the 14 monitored appliances (seven appliances in House 1 and seven appliances in House 2) were accurately detected. The processing at the edge device reduced unnecessary data transmission while maintaining reliable detection performance, which is essential for enabling responsive and scalable NILM systems.

The use of unsupervised learning techniques enabled the automatic creation of a tailored appliance-specific dataset, avoiding the need to rely on public datasets that, for real-world implementations, would require hundreds of different appliances to encompass the diversity found in residential environments. Such extensive datasets typically demand the use of highly complex deep learning models for training, increasing computational requirements and reducing deployment practicality. By focusing on the actual appliances present in the monitored households, the system reduced model complexity while maintaining classification relevance. This strategy not only adapted to

the unique load characteristics of each environment but also enhanced the replicability of the project, allowing similar systems to be deployed in other settings with minimal retraining effort.

Furthermore, the comparative evaluation of three deep learning architectures for time series classification—namely, a 1D Convolutional Neural Network, a Long Short-Term Memory network, and a WaveNet—showed that all models achieved accuracy, precision, recall, and F1-score values above 97%. Among them, the LSTM model emerged as the most suitable option, offering a better balance between predictive performance and computational efficiency. This advantage allowed the classification module to operate effectively in both edge and cloud-based processing workflows without compromising operational speed.

In addition, the development of a web application with an integrated classification module provided a platform for visualizing detected events and their corresponding classifications in real-time. The IoT architecture implemented enabled the processing of streaming data such that, on average, an event could be classified in approximately 1.05 s from the moment the three signatures of the event were sent from the smart meter to the point when the model inferred an output on the server. This capability supported timely feedback to the user, enhancing the practicality of energy monitoring and management applications.

Moreover, the deployment of the proposed NILM system in two different residential environments demonstrated its robustness and adaptability. Across diverse electrical usage patterns, the system maintained classification accuracy over 95% and stable operation, confirming its applicability for real-world energy monitoring scenarios and its potential for large-scale deployment.

Finally, the experimental results strongly validate the proposed hypothesis, demonstrating that the developed IoT-based NILM system—integrating real-time event detection and unsupervised learning to build a self-adaptive appliance dataset—can achieve accurate and scalable appliance-level disaggregation in real residential environments. The system surpassed the stated performance targets, achieving over 95 % accuracy in both event detection and classification, which sets a solid benchmark for practical NILM implementations. Moreover, the average response time of only 1.05 s from event signature transmission to server-side inference is well below the 3-second threshold defined in the objectives, ensuring that the system operates within real-time requirements for IoT-based NILM in residential environments. These outcomes were achieved without relying on public datasets or high-frequency parameters, confirming the feasibility, efficiency, and robustness of the proposed approach across diverse real-

world conditions.

6.2 Future Work

Future research could explore the integration of non-intrinsic contextual parameters—defined as factors external to the electrical signatures themselves that influence appliance usage—into the classification process. Information such as the time of day when an appliance is switched on or off, the number of times it operates within a given period, and seasonal or day-of-week usage patterns could complement electrical signatures. Incorporating these parameters into the deep learning models may improve robustness, particularly when distinguishing between appliances with similar electrical characteristics. This assumption is based on the rationale that additional contextual features provide supplementary information that could help the model differentiate appliance usage patterns, potentially enhancing detection accuracy through context-aware classification strategies.

Also, an important direction involves the development of a new smart meter architecture capable of training and deploying deep learning models directly on the device. This would require optimization techniques such as model quantization, pruning, and on-device transfer learning to fit the constraints of embedded hardware. By enabling the classification algorithm to operate entirely within the smart meter, the system could reduce latency, enhance privacy, and maintain reliable operation in environments with limited or intermittent connectivity.

Another area of interest is the extension of the system to handle complex appliances with multi-phase operational cycles, such as washing machines or dishwashers. These devices involve multiple subsystems that operate sequentially or simultaneously, producing diverse electrical patterns over a single usage cycle. In addition, expanding the event detection and classification framework to manage simultaneous events from multiple appliances would improve performance in high-activity households and better reflect real-world operating conditions.

Further advancements could also involve multi-label classification and sequence-to-sequence modeling to address overlapping and interdependent appliance events. This would allow the system to capture the dynamic interactions between multiple devices, moving beyond isolated event recognition toward a more comprehensive understanding of household energy usage.

Bibliography

[1] F Abate, M Carratu, C Liguori, and V Paciello. A low cost smart power meter for IoT. *Measurement*, 136:59—-66., 2019.

[2] M Aboelmaged, Y Abdelghani, and M A A E Ghany. Wireless IoT based metering system for energy efficient smart cites. In *2017 29th International Conference on Microelectronics (ICM)*, pages 1–4, 2017.

[3] Prakash Pawar and Panduranga Vittal K. Design and development of advanced smart energy management system integrated with IoT framework in smart grid environment. *Journal of Energy Storage*, 25, oct 2019.

[4] Qie Sun, Hailong Li, Zhanyu Ma, Chao Wang, Javier Campillo, Qi Zhang, Fredrik Wallin, and Jun Guo. A Comprehensive Review of Smart Energy Meters in Intelligent Energy Networks. *IEEE Internet of Things Journal*, 3(4):464–479, 2016.

[5] Wilson L Rodrigues Junior, Fabbio A S Borges, Artur F Da, S Veloso, Ricardo De A L Rabêlo, and Joel J P C Rodrigues. Low voltage smart meter for monitoring of power quality disturbances applied in smart grid. 2019.

[6] Damminda Alahakoon and Xinghuo Yu. Smart Electricity Meter Data Intelligence for Future Energy Systems: A Survey. *IEEE Transactions on Industrial Informatics*, 12(1):425–436, 2016.

[7] Danielly B. Avancini, Joel J.P.C. Rodrigues, Simion G.B. Martins, Ricardo A.L. Rabêlo, Jalal Al-Muhtadi, and Petar Solic. Energy meters evolution in smart grids: A review. *Journal of Cleaner Production*, 217:702–715, 2019.

[8] Rosario Morello, Claudio De Capua, Gaetano Fulco, and Subhas Chandra Mukhopadhyay. A smart power meter to monitor energy flow in smart grids: The role of advanced sensing and iot in the electric grid of the future. *IEEE Sensors Journal*, 17(23):7828–7837, 2017.

[9] I Abubakar, SN Khalid, MW Mustafa, Hussain Shareef, and M Mustapha. Application of load monitoring in appliances' energy management-A review. 43000(1), 2016.

[10] Xiangyu Kong, Siqiong Zhang, Bowei Sun, Qun Yang, Shupeng Li, and Shijian Zhu. Research on Home Energy Management Method for Demand Response Based on Chance-Constrained Programming. *Energies*, 13:2790, 2020.

[11] E. J. Aladesanmi and K. A. Folly. Overview of non-intrusive load monitoring and identification techniques. In *IFAC-PapersOnLine*, volume 48, pages 415–420. Elsevier, jan 2015.

[12] Xin Shi, Hao Ming, Srinivas Shakkottai, Le Xie, and Jianguo Yao. Nonintrusive load monitoring in residential households with low-resolution data. 2019.

[13] Sheharyar Khan, Ahmad Farhan Latif, and Sarmad Sohaib. Low-cost real-time non-intrusive appliance identification and controlling through machine learning algorithm. In *2018 International Symposium on Consumer Technologies, ISCT 2018*, number Ilm, pages 32–36. IEEE, 2018.

[14] George W Hart. Nonintrusive appliance load monitoring. *Proceedings of the IEEE*, 80(12):1870–1891, 1992.

[15] Hafiz Khurram Iqbal, Farhan Hassan Malik, Aoun Muhammad, Muhammad Ali Qureshi, Muhammad Nawaz Abbasi, and Abdul Rehman Chishti. A critical review of state-of-the-art non-intrusive load monitoring datasets. *Electric Power Systems Research*, 192:106921, 2021.

[16] Youngwook Kim, Seongbae Kong, Rakkyung Ko, and Sung Kwan Joo. Electrical event identification technique for monitoring home appliance load using load signatures. *Digest of Technical Papers - IEEE International Conference on Consumer Electronics*, pages 296–297, 2014.

[17] Corinne Belley, Sebastien Gaboury, Bruno Bouchard, and Abdenour Bouzouane. Nonintrusive system for assistance and guidance in smart homes based on electrical devices identification. *Expert Systems with Applications journal*, 42, 2015.

[18] Shunfu Lin, Lunjia Zhao, Fangxing Li, Qingqiang Liu, Dongdong Li, and Yang Fu. A nonintrusive load identification method for residential applications based on quadratic programming. *Electric Power Systems Research*, 133:241–248, 2016.

[19] Yuzhou Zhou, Qiaozhu Zhai, Xuan Li, and Yafei Yang. A method for recognizing electrical appliances based on active load demand in a house/office environment. In *Proceedings - 2017 Chinese Automation Congress, CAC 2017*, volume 2017-Janua, pages 3584–3589, 2017.

[20] Emmanuel Guillén García, Luis Morales-Velazquez, Angel L Zorita-Lamadrid, Oscar Duque-Perez, Roque Alfredo Osornio-Rios, and Rene de Jesús Romero-Troncoso. Identification of the electrical load by C-means from non-intrusive monitoring of electrical signals in non-residential buildings. 2018.

[21] Zhuang Zheng, Hainan Chen, and Xiaowei Luo. A supervised event-based non-intrusive load monitoring for non-linear appliances. *Sustainability (Switzerland)*, 10(4):1–28, 2018.

[22] Chunjiao Yu, Pengfei Chen, Xiaosheng Liu, Liang Zhao, Ming Han, and Yousu Yao. Design of a Smart Socket Functioned with Electrical Appliance Identification. In *2019 22nd International Conference on Electrical Machines and Systems, ICEMS 2019*, pages 1–5. IEEE, 2019.

[23] Lorena R. Morais and Adriana R.G. Castro. Competitive Autoassociative Neural Networks for Electrical Appliance Identification for Non-Intrusive Load Monitoring. *IEEE Access*, 7:111746–111755, 2019.

[24] Yong Xiao, Yue Hu, Hengjing He, Dongguo Zhou, Yun Zhao, and Wenshan Hu. Non-Intrusive Load Identification Method Based on Improved KM Algorithm. *IEEE Access*, 7:151368–151377, 2019.

[25] Shirantha Welikala, Chinthaka Dinesh, Mervyn Parakrama B. Ekanayake, Roshan Indika Godaliyadda, and Janaka Ekanayake. Incorporating Appliance Usage Patterns for Non-Intrusive Load Monitoring and Load Forecasting. *IEEE Transactions on Smart Grid*, 10(1):448–461, jan 2019.

[26] Zhaoyuan Fang, Dongbo Zhao, Chen Chen, Yang Li, and Yutin Tian. Nonintrusive Appliance Identification with Appliance-Specific Networks. *IEEE Transactions on Industry Applications*, 56(4):3443–3452, 2020.

[27] Radu-Casian Mihailescu, David Hurtig, and Charlie Olsson. End-to-end anytime solution for appliance recognition based on high-resolution current sensing with few-shot learning. *Internet of Things*, 11:100263, 2020.

[28] Emre Akarslan and Rasim Do. A novel approach for residential load appliance identification. 2020.

[29] Xin Wu, Dian Jiao, and Lan You. Nonintrusive on-site load-monitoring method with self-adaption. *Electrical Power and Energy Systems*, 119, 2020.

[30] Zeynep Duygu Tekler, Raymond Low, Yuren Zhou, Chau Yuen, Lucienne Blessing, and Costas Spanos. Near-real-time plug load identification using low-frequency power data in office spaces: Experiments and applications. 2020.

[31] Wen Fan, Qing Liu, Ali Ahmadpour, and Saeed Gholami Farkoush. Multi-objective non-intrusive load disaggregation based on appliances characteristics in smart homes. *Energy Reports*, 7:4445–4459, 2021.

[32] Chen Chen, Pinghang Gao, Jiange Jiang, Hao Wang, Pu Li, and Shaohua Wan. A deep learning based non-intrusive household load identification for smart grid in China. *Computer Communications*, 177:176–184, 2021.

[33] Yassine Himeur, Abdullah Alsalemi, Faycal Bensaali, and Abbes Amira. Smart non-intrusive appliance identification using a novel local power histogramming descriptor with an improved k-nearest neighbors classifier. 2021.

[34] Md Tanvir Ahammed, Md Mehedi Hasan, Md Shamsul Arefin, Md Rafiqul Islam, Md Aminur Rahman, Eklas Hossain, and Md Tanvir Hasan. Real-Time Non-Intrusive Electrical Load Classification over IoT Using Machine Learning. *IEEE Access*, 9:115053–115067, 2021.

[35] Patricia Franco, Jose Manuel Martinez, Young Chon Kim, and Mohamed A. Ahmed. IoT Based Approach for Load Monitoring and Activity Recognition in Smart Homes. *IEEE Access*, 9:45325–45339, 2021.

[36] Bo Yin, Liwen Zhao, Xianqing Huang, Ying Zhang, and Zehua Du. Research on non-intrusive unknown load identification technology based on deep learning. 2021.

[37] Yanzhen Li, Haixin Wang, Junyou Yang, Kang Wang, and Guanqiu Qi. A non-intrusive load monitoring algorithm based on multiple features and decision fusion. *Energy Reports*, 7:1555–1562, nov 2021.

[38] Christos L. Athanasiadis, Theofilos A. Papadopoulos, and Dimitrios I. Doukas. Real-time non-intrusive load monitoring: A light-weight and scalable approach. *Energy and Buildings*, 253:111523, dec 2021.

[39] Zejian Zhou, Yingmeng Xiang, Hao Xu, Yishen Wang, and Di Shi. Unsupervised Learning for Non-intrusive Load Monitoring in Smart Grid Based on Spiking Deep Neural Network. *JOURNAL OF MODERN POWER SYSTEMS AND CLEAN ENERGY*, pages 1–11, 2021.

[40] Yu-Hsiu Lin. Trainingless multi-objective evolutionary computing-based nonintrusive load monitoring: Part of smart-home energy management for demand-side management. *Journal of Building Engineering*, 33:101601, 2021.

[41] Biqi Liu, Fuxiang Zhang, Xi Li, He Wu, and Libo Xu. Power load identification based on Long-and-Short-Term Memory network and Affinity Propagation clustering algorithm. *Energy Reports*, 8:1137–1144, jul 2022.

[42] Miao Yu, Bingnan Wang, Lingxia Lu, Zhejing Bao, and Donglian Qi. Non-Intrusive Adaptive Load Identification Based on Siamese Network. *IEEE Access*, 10:11564–11573, 2022.

[43] Bo Liu, Jinhao Zheng, Wenpeng Luan, Fenglei Chang, Bochao Zhao, and Zishuai Liu. Enhanced nilm load pattern extraction via variable-length motif discovery. *International Journal of Electrical Power & Energy Systems*, 152:109207, 2023.

[44] Johannes Winkler, Hafsa Bousbiat, Stefan Jost, and Wilfried Elmenreich. Energy disaggregation with nilm on a raspberry pi with smart-metering extension. pages 191–195. Institute of Electrical and Electronics Engineers Inc., 2023.

[45] Noor El Deen M. Mohamed, Mohamed M. El-Dakroury, Abdulrahman A. Ibrahim, and George A. Nfady. Iot next generation smart grid meter (ngsm) for on-edge household appliances detection based on deep learning and embedded linux. pages 410–415. Institute of Electrical and Electronics Engineers Inc., 2023.

[46] Hala Najmeddine, Khalil El Khamlichi Drissi, Christophe Pasquier, Claire Faure, Kamal Kerroum, Alioune Diop, Thierry Jouannet, and Michel Michou. State of art on load monitoring methods. In *PECon 2008 - 2008 IEEE 2nd International Power and Energy Conference*, number PECon 08, pages 1256–1258, 2008.

[47] Christos Timplalexis, Stelios Krinidis, Dimosthenis Ioannidis, and Dimitrios Tzovaras. NILM applications: Literature review of learning approaches, recent developments and challenges. *Energy & Buildings*, 2022.

[48] Lei Yan, Mehrdad Sheikholeslami, Wenlong Gong, Wei Tian, and Zuyi Li. Challenges for real-world applications of nonintrusive load monitoring and

opportunities for machine learning approaches. *The Electricity Journal journal*, 2022.

[49] Hasan Rafiq, Prajowal Manandhar, Edwin Rodriguez-Ubinas, Omer Ahmed Qureshi, and Themis Palpanas. A review of current methods and challenges of advanced deep learning-based non-intrusive load monitoring (nilm) in residential context, 2 2024.

[50] Qingquan Luo, Tao Yu, Minhang Liang, Zhenning Pan, Wenlong Guo, and Xiaolei Hu. Review of advances in scaling non-intrusive load monitoring for real-world applications. *Applied Energy*, 398, 11 2025.

[51] David Cruz-Rangel, Carlos Ocampo-Martinez, and Javier Diaz-Rozo. Online non-intrusive load monitoring: A review. *Energy Nexus*, 17:100348, 3 2025.

[52] Zhuang Zheng, Hainan Chen, and Xiaowei Luo. A supervised event-based non-intrusive load monitoring for non-linear appliances. *Sustainability*, 10(4):1001, 2018.

[53] Bo Yin, Liwen Zhao, Xianqing Huang, Ying Zhang, and Zehua Du. Research on non-intrusive unknown load identification technology based on deep learning. *International Journal of Electrical Power & Energy Systems*, 131:107016, 2021.

[54] Everton Luiz De Aguiar, Lucas da Silva Nolasco, André Eugenio Lazzaretti, Daniel Rodrigues Pipa, and Heitor Silvério Lopes. St-nilm: A wavelet scattering-based architecture for feature extraction and multi-label classification in nilm signals. *IEEE Sensors Journal*, 2024.

[55] T. Karthick, S. Charles Raja, J. Jeslin Drusila Nesamalar, and K. Chandrasekaran. Design of IoT based smart compact energy meter for monitoring and controlling the usage of energy and power quality issues with demand side management for a commercial building. *Sustainable Energy, Grids and Networks*, 26:100454, jun 2021.

[56] Shishir Muralidhara, Niharika Hegde, and Rekha PM. An internet of things-based smart energy meter for monitoring device-level consumption of energy. *Computers and Electrical Engineering*, 87:106772, oct 2020.

[57] Electricity generation. `https://ourworldindata.org/grapher/electricity-generation?tab=chart&country=~OWID_WRL` [Accessed: 2025-05-09].

[58] Energy production and consumption – our world in data. https://ourworldindata.org/energy-production-consumption [Accessed: 2025-05-09].

[59] M. Kuzlu, M. Pipattanasomporn, and S. Rahman. Review of communication technologies for smart homes/building applications. In *Proceedings of the 2015 IEEE Innovative Smart Grid Technologies - Asia, ISGT ASIA 2015*, number November, 2016.

[60] Suryalok Dash and N C Sahoo. Electric Power Systems Research Electric energy disaggregation via non-intrusive load monitoring: A state-of-the-art systematic review. *Electric Power Systems Research*, 213:108673, 2022.

[61] R V A Monteiro, J C R De Santana, R F S Teixeira, A S Bretas, R Aguiar, and C E P Poma. Non-intrusive load monitoring using artificial intelligence classifiers: Performance analysis of machine learning techniques. *Electric Power Systems Research*, 198:107347, 2021.

[62] Michael A. Devlin and Barry P. Hayes. Load identification and classification of activities of daily living using residential smart meter data. In *2019 IEEE Milan PowerTech, PowerTech 2019*, pages 1–6. IEEE, 2019.

[63] DrMSundararajan Murugaiyan. WATEERFALLVs V-MODEL Vs AGILE: A COMPARATIVE STUDY ON SDLC. *International Journal of Information Technology and Business Management*, 2(1), 2012.

[64] Wesley A Souza, Augusto M S Alonso, Thais B Bosco, Fernando D Garcia, Flavio A S Gonçalves, and Fernando P Marafão. Selection of features from power theories to compose nilm datasets. 2022.

[65] Pervez Hameed Shaikh, Nursyarizal Bin Mohd Nor, Perumal Nallagownden, Irraivan Elamvazuthi, and Taib Ibrahim. A review on optimized control systems for building energy and comfort management of smart sustainable buildings. *Renewable and Sustainable Energy Reviews*, 34:409–429, 2014.

[66] Antonio Ridi, Christophe Gisler, and Jean Hennebert. A survey on intrusive load monitoring for appliance recognition. In *2014 22nd international conference on pattern recognition*, pages 3702–3707. IEEE, 2014.

[67] Hermann Kopetz and Wilfried Steiner. *Real-Time Systems*. Computer Science, Computer Science (R0). Springer Cham, Cham, 3 edition, September 2022.

Copyright © The Editor(s) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2022.

[68] Roberto Bonfigli, Emanuele Principi, Marco Fagiani, Marco Severini, Stefano Squartini, and Francesco Piazza. Non-intrusive load monitoring by using active and reactive power in additive factorial hidden markov models. *Applied Energy*, 208:1590–1607, 2017.

[69] Bruna M. Mulinari, Daniel P. de Campos, Clayton H. da Costa, Hellen C. Ancelmo, André E. Lazzaretti, Elder Oroski, Carlos R. E. Lima, Douglas P. B. Renaux, Fabiana Pottker, and Robson R. Linhares. A new set of steady-state and transient features for power signature analysis based on v-i trajectory. In *2019 IEEE PES Innovative Smart Grid Technologies Conference - Latin America (ISGT Latin America)*, pages 1–6, 2019.

[70] Patrick Huber, Alberto Calatroni, Andreas Rumsch, and Andrew Paice. Review on deep neural networks applied to low-frequency nilm. *Energies*, 14, 2021.

[71] Dawei He, Liang Du, Yi Yang, Ronald Harley, and Thomas Habetler. Front-end electronic circuit topology analysis for model-driven classification and monitoring of appliance loads in smart buildings. *IEEE Transactions on Smart Grid*, 3:2286–2293, 2012.

[72] HY Lam, K Ting, WK Lee, and G Fung. An analytical understanding on voltage-current curve of electrical load. In *International Conference on Electrical Engineering (ICEE)*, pages 1–6, 2006.

[73] Charles K Alexander. *Fundamentals of electric circuits*. McGraw-Hill,, 2013.

[74] John G Webster. *The Measurement, Instrumentation, and Sensors: Handbook*. Springer Science & Business Media, 2da edition, 1999.

[75] J David Irwin and R Mark Nelms. *Basic engineering circuit analysis*. John Wiley & Sons, 2020.

[76] Robert W Erickson and Dragan Maksimovic. *Fundamentals of power electronics*. Springer Science & Business Media, 2007.

[77] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Pearson, 2016.

[78] Luisa Mich. Artificial intelligence and machine learning. *Handbook of e-Tourism*, pages 1–21, 2020.

[79] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2020.

[80] Gopinath Rebala, Ajay Ravi, and Sanjay Churiwala. *An introduction to machine learning*. Springer, 2019.

[81] Seth Weidman. *Deep learning from scratch: Building with python from first principles*. O'Reilly Media, 2019.

[82] Deep Learning. Deep learning-goodfellow. *Nature*, 26(7553):436, 2016.

[83] Yordan P Raykov, Alexis Boukouvalas, Fahd Baig, and Max A Little. What to do when k-means clustering fails: a simple yet principled alternative algorithm. *PloS one*, 11(9):e0162259, 2016.

[84] Simon Haykin. *Neural Networks and Learning Machines*. Pearson Education, 3rd edition, 2009.

[85] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Sebastopol, CA, 2nd edition, 2019.

[86] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 33(12):6999–7019, 2021.

[87] Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

[88] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019.

[89] Sebastian Raschka and Vahid Mirjalili. *Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2*. Packt publishing ltd, 2019.

[90] Ovidiu Vermesan, Peter Friess, and Peter Friess. *Internet of Things: Global technological and societal trends*. River Publishers Aalborg, Denmark, 1ra edition, 2011.

[91] R van Kranenburg and S Dodson. *The Internet of Things: A Critique of Ambient Technology and the All-seeing Network of RFID*. Network notebooks. Institute of Network Cultures, 2008.

[92] Arshdeep Bahga and Vijay Madisetti. *Internet of Things: A hands-on approach*. Vpt, 1ra edition, 2014.

[93] Ibrahim Mashal, Osama Alsaryrah, Tein-Yaw Chung, Cheng-Zen Yang, Wen-Hsing Kuo, and Dharma P Agrawal. Choices for interaction with things on Internet and underlying issues. *Ad Hoc Networks*, 28:68–90, 2015.

[94] Mahesh Kalmeshwar and Assoc. Professor Dr. Nandini Prasad K S. Internet Of Things: Architecture,Issues and Applications. *International Journal of Engineering Research and Applications*, 07(06):85–88, 2017.

[95] Fatima Hussain. *Internet of Things Building Blocks and Business Models*. Springer International Publishing, 1ra edition, 2017.

[96] P. P. Ray. A survey on Internet of Things architectures. *Journal of King Saud University - Computer and Information Sciences*, 30(3):291–319, 2018.

[97] Dina Gamal Darwish and Elroda Square. Improved Layered Architecture for Internet of Things. *International Journal of Computing Academic Research*, 4(4):214–223, 2015.

[98] Mamoona Majid, Shaista Habib, Abdul Rehman Javed, Muhammad Rizwan, Gautam Srivastava, Thippa Reddy Gadekallu, and Jerry Chun-Wei Lin. Applications of wireless sensor networks and internet of things frameworks in the industry revolution 4.0: A systematic literature review. *Sensors*, 22(6):2087, 2022.

[99] Yang Shuang-Hua. *Wireless sensor networks: Principles, Design and Applications*. Springer-Verlag London, 1ra edition, 2014.

[100] Simone Cirani, Gianluigi Ferrari, Marco Picone, and Luca Veltri. *Internet of Things: Architectures, Protocols and Standards*. John Wiley & Sons, 1ra edition, 2019.

[101] Diksha Chawla and Pawan Singh Mehra. A roadmap from classical cryptography to post-quantum resistant cryptography for 5g-enabled iot: Challenges, opportunities and solutions. *Internet of Things*, 24:100950, 2023.

[102] Nuno Miguel Carvalho Galego, Rui Miguel Pascoal, and Pedro Ramos Brandão. Ipv6 in iot. In *International Conference on Management, Tourism and Technologies*, pages 89–94. Springer, 2023.

[103] Taief Alaa Al-Amiedy, Mohammed Anbar, Bahari Belaton, Abdullah Ahmed Bahashwan, Iznan Husainy Hasbullah, Mohammad Adnan Aladaileh, and Ghada AL Mukhaini. A systematic literature review on attacks defense mechanisms in rpl-based 6lowpan of internet of things. *Internet of Things*, 22:100741, 2023.

[104] Sukjun Hong, Jinkyu Kang, and Soonchul Kwon. Performance comparison of http, https, and mqtt for iot applications. *International journal of advanced smart convergence*, 12(1):9–17, 2023.

[105] Z Shelby, K Hartke, and C Borman. The Constrained Application Protocol (CoAP). Technical report, IETF, 2014.

[106] Richard Coppen, Andrew Banks, Ed Briggs, and Ken Borgendale. MQTT Version 5.0 OASIS Standard. Technical report, OASIS, 2019.

[107] Maysam Alkwiefi et al. M2m protocols: An overview on lwm2m and xmpp machine-to-machine protocols in iot context. In *2024 IEEE/ACIS 24th International Conference on Computer and Information Science (ICIS)*, pages 209–215. IEEE, 2024.

[108] Chnar Mustafa Mohammed and Subhi RM Zeebaree. Sufficient comparison among cloud computing services: Iaas, paas, and saas: A review. *International Journal of Science and Business*, 5(2):17–30, 2021.

[109] Shilpy Sharma, David A. Swayne, and Charlie Obimbo. Trend analysis and change point techniques: a survey. *Energy, Ecology and Environment*, 1:123–130, 6 2016.

[110] Masoomeh Zameni, Amin Sadri, Zahra Ghafoori, Masud Moshtaghi, Flora D. Salim, Christopher Leckie, and Kotagiri Ramamohanarao. Unsupervised online change point detection in high-dimensional time series. *Knowledge and Information Systems*, 62:719–750, 2 2020.

[111] Jie Li, Paul Fearnhead, Piotr Fryzlewicz, and Tengyao Wang. Automatic change-point detection in time series via deep learning. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 86:273–285, 4 2024.

[112] Mohamed Nait-Meziane, Philippe Ravier, Karim Abed-Meraim, Guy Lamarque, Jean-Charles Le Bunetel, and Yves Raingeaud. Electrical transient modeling for appliance characterization. *EURASIP Journal on Advances in Signal Processing*, 2019(1):55, 2019.

[113] Dipti Theng and Kishor K Bhoyar. Feature selection techniques for machine learning: a survey of more than two decades of research. *Knowledge and Information Systems*, 66(3):1575–1637, 2024.

[114] Dongying Chen, Hao Zhang, Lingyan Lin, Zilong Zhang, Jian Zeng, Lu Chen, and Xiaogang Chen. Auto-encoder design based on the 1d-vd-cnn model for the detection of honeysuckle from unknown origin. *Journal of Pharmaceutical and Biomedical Analysis*, 234:115572, 2023.

[115] Roberto Medico, Leen De Baets, Jingkun Gao, Suman Giri, Emre Kara, Tom Dhaene, Chris Develder, Mario Berges, and Dirk Deschrijver. A voltage and current measurement dataset for plug load appliance identification in households. *Scientific data*, 7(1):49, 2020.

Appendix A

Academic Products

A.1 Article Title: *Design and implementation of an automatic and self-adaptive NILM system using unsupervised learning and an IoT platform*

Contents lists available at ScienceDirect

Electric Power Systems Research

journal homepage: www.elsevier.com/locate/epsr

Design and implementation of an automatic and self-adaptive NILM system using unsupervised learning and an IoT platform

Omar Munoz, Adolfo Ruelas *, Pedro F. Rosales-Escobedo, Jorge E. Ibarra-Esquer, Ruben A. Reyes-Zamora, Alexis Acuña, Alejandro Suastegui

Facultad de Ingeniería, Universidad Autónoma de Baja California, Blvd. Benito Juárez S/N, Mexicali 21280, Mexico

ARTICLE INFO

Keywords:
NILM
Load monitoring
Smart grid
IoT
Smart meter

ABSTRACT

The consumption of electricity has been increasing significantly because of the rising population, the impacts of climate change, rapid urbanization, and the widespread adoption of electronic devices. Smart grids are essential for efficient energy management, integrating renewable resources, optimizing power systems, and addressing different power demands. Non-Intrusive Load Monitoring (NILM) enhances their effectiveness by disaggregating overall consumption into appliance-specific data, enabling precise monitoring and control. This paper presents the design and implementation of an IoT NILM system that incorporates a custom smart meter with a real-time event detection algorithm, a light IoT framework and an unsupervised learning approach for creating a self-adaptive dataset. Through RMS current and active/reactive power transient signals obtained from the event detection algorithm working in a real-world scenario, a 1D-CNN autoencoder model extracts characteristics from the time series and they are used to feed a K-means algorithm with auto K selection based on silhouette score. The clusters of the K-means achieved an accuracy of 99.22% in classification and became the dataset tailored for end users. For demonstrating the effectiveness and usability of the created dataset, a 1D-CNN classification model was implemented, achieving an accuracy of 98.53%.

Figure A.1: Cover page of the article published in *Electric Power Systems Research* (Elsevier). DOI: 10.1016/j.epsr.2024.111376

A.2 Article Title: *Development of an IoT smart energy meter with power quality features for a smart grid architecture*

Contents lists available at ScienceDirect

Sustainable Computing: Informatics and Systems

journal homepage: www.elsevier.com/locate/suscom

Development of an IoT smart energy meter with power quality features for a smart grid architecture

Omar Munoz, Adolfo Ruelas *, Pedro F. Rosales-Escobedo, Alexis Acuña, Alejandro Suastegui, Fernando Lara, Ruben A. Reyes-Zamora, Angel Rocha

Facultad de Ingeniería, Universidad Autónoma de Baja California, Blvd. Benito Juárez S/N, Mexicali 21280, Mexico

ARTICLE INFO

Keywords:
IoT
Smart meter
Power meter
Smart grid

ABSTRACT

Electricity consumption has been intensifying due to population growth, climate change, urbanization, and the growing use of electronic devices, which are increasingly non-linear loads that cause poor power quality conditions. The trend of the Internet of Things has led to the creation of devices that encourage the efficient and effective utilization of electrical power. This in turn facilitates the development of modern power distribution structures such as smart grids. Consequently, this paper presents in detail the design, construction, and validation of a three-phase IoT smart meter intended to form part of the end-user demand side of a smart grid. The compact embedded system, with a manufacturing cost below $80 USD, features a unique electronic design that enables its installation in any load center and employs a straightforward IoT structure that includes WiFi technology for Internet communication. Also, a deployed web application was developed specifically to display the smart meter measurements. Unlike other smart meters, the proposed meter not only provides the amount of active energy consumption, but total and fundamental RMS current and voltage, active, reactive, and apparent power, reactive energy, power factor, and some power quality parameters such as, line frequency, amplitude of 64 current harmonics, and total harmonic distortion. Additionally, this study shows that the prototype achieves an absolute error of less than 1% in all its measurements. Finally, real-life applications of the developed device are demonstrated in residential environments.

Figure A.2: Cover page of the article published in *Sustainable Computing: Informatics and Systems* (Elsevier). DOI:10.1016/j.suscom.2024.100990

A.3 Article Title: *Design and Development of an IoT Smart Meter with Load Control for Home Energy Management Systems*

sensors

MDPI

Article

Design and Development of an IoT Smart Meter with Load Control for Home Energy Management Systems

Omar Munoz, Adolfo Ruelas *, Pedro Rosales, Alexis Acuña, Alejandro Suastegui and Fernando Lara

Facultad de Ingeniería, Universidad Autónoma de Baja California, Blvd. Benito Juárez S/N, Mexicali 21280, Mexico

* Correspondence: ruelasa@uabc.edu.mx

Abstract: Electricity consumption is rising due to population growth, climate change, urbanization, and the increasing use of electronic devices. The trend of the Internet of Things has contributed to the creation of devices that promote the thrift and efficient use of electrical energy. Currently, most projects relating to this issue focus solely on monitoring energy consumption without providing relevant parameters or switching on/off electronic devices. Therefore, this paper presents in detail the design, construction, and validation of a smart meter with load control aimed at being part of a home energy management system. With its own electronic design, the proposal differs from others in many aspects. For example, it was developed using a simple IoT architecture with in-built WiFi technology to enable direct connection to the internet, while at the same time being big enough to be part of standardized electrical enclosures. Unlike other smart meters with load control, this one not only provides the amount of energy consumption, but rms current and voltage, active, reactive, and apparent power, reactive energy, and power factor—parameters that could be useful for future studies. In addition, this work presents evidence based on experimentation that the prototype in all its readings achieves an absolute percentage error of less than 1%. A real-life application of the device was also demonstrated in this document by measuring different appliances and switching them on/off manually and automatically using a web-deployed application.

Keywords: smart meter; power meter; internet of things; load control; energy meter; smart socket

check for updates

Citation: Munoz, O.; Ruelas, A.; Rosales, P.; Acuña, A.; Suastegui, A.; Lara, F. Design and Development of an IoT Smart Meter with Load Control for Home Energy Management Systems. *Sensors* **2022**, *22*, 7536. https://doi.org/10.3390/s22197536

Figure A.3: Cover page of the article published in *Sensors* (MDPI). DOI: 10.3390/s22197536

A.4 Proceedings Article Title: *A WaveNet-Based IoT System for Non-Intrusive Load Monitoring*

Figure A.4: Certificate of paper acceptance and publication for the article presented at the 10th International Conference on Information and Communication Technology for Intelligent Systems (ICTIS 2025), New York, USA. The paper was included in the book *ICT for Intelligent Systems: Proceedings of ICTIS 2025, Volume 13*, ISBN: 978-981-95-1352-9.

Appendix B

Datasets

B.1 House 1 and House 2 signatures

https://github.com/omarmuoz-lab/NILM.git

Appendix C

Smart Meter

C.1 PCB Design

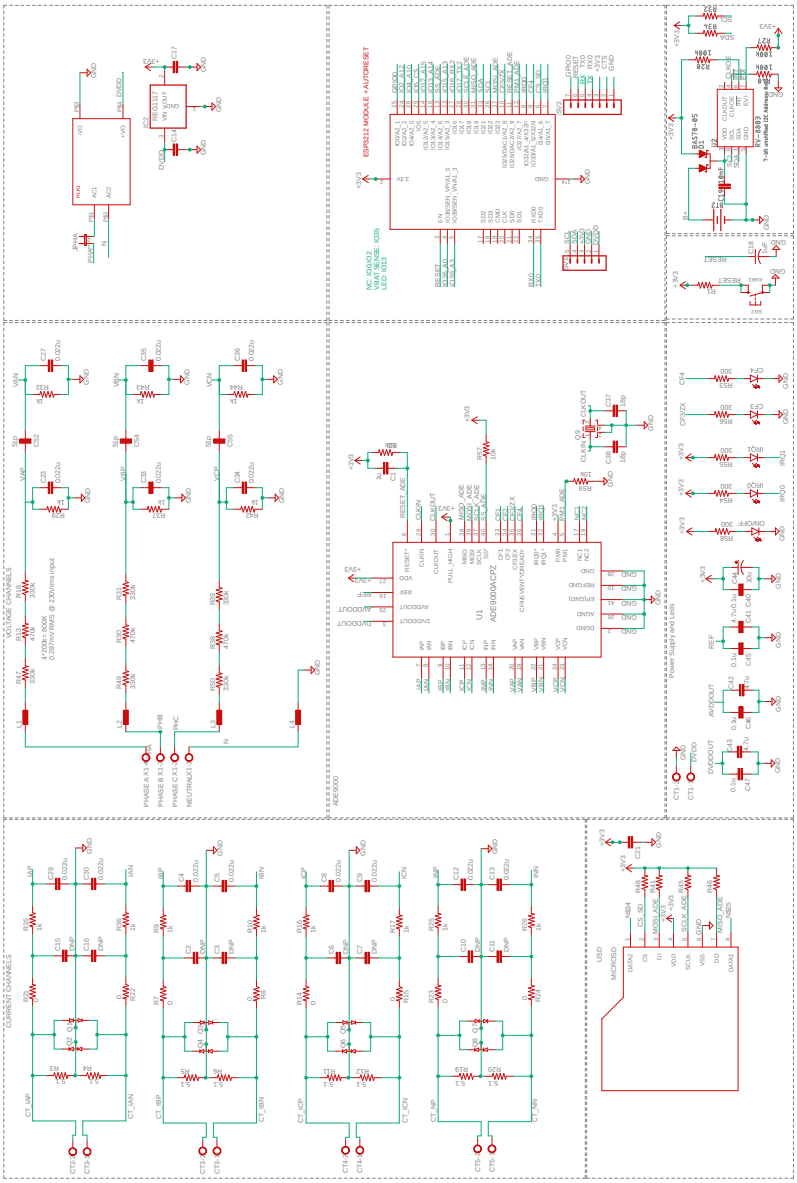


Figure C.1: Schematic diagram of the designed IoT smart meter.

C.2 PCB Design

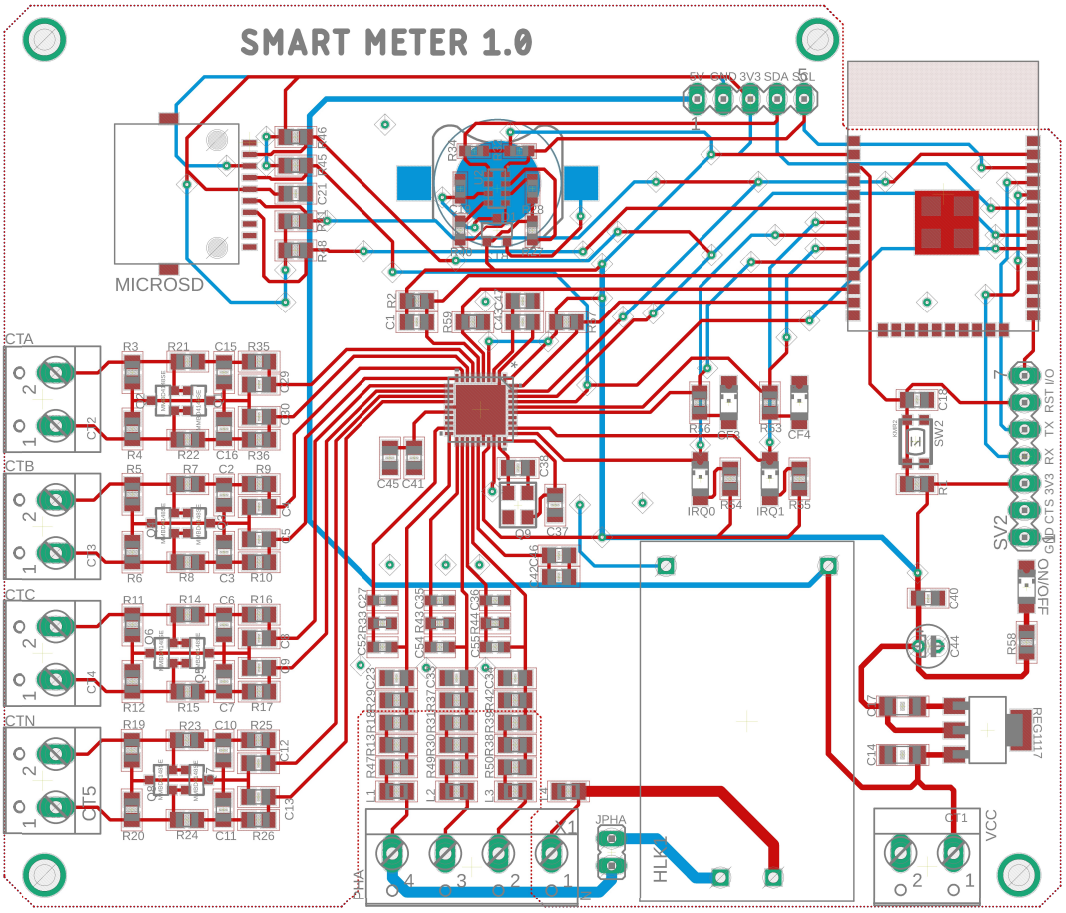


Figure C.2: PCB of the designed IoT smart meter.

C.3 Firmware

https://github.com/omarmuoz-lab/Smart-Meter/tree/main/Firmware